MSX

dBASE II

VERSÃO 1.0 - 1987

DATALOGICA

ASHTON-TATE

dBASE II PLUS MSX é um software nacional escrito em ASSEMBLER para processadores 8080 e compatíveis, desenvolvido pela equipe de software básico da DATALOGICA, homologado tecnicamente e considerado 100% compatível pelos técnicos da ASHTON-TATE. Além disto, possui precisão numérica de 16 dígitos e um vasto número de funções implementadas que justificam o nome PLUS.

Av. Paulista, 2028 - 16º
Tlf.: (011) 283-0355
Tlx.: (11) 32-645 DTLG BR
CEP 01310 - São Paulo - SP

Av. Rio Branco, 177 - 15º
Tlf.: (021) 220-2242
Tlx.: (21) 31-615 DTLG BR
CEP 20040 - Rio de Janeiro - RJ

R. Mostardeiro, 333 Conj. 408 - 4º
Tlf.: (0512) 22-9710
CEP 90410 - Porto Alegre - RS

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO

CAPÍTULO 2

REFERÊNCIA RÁPIDA

CAPÍTULO 3

ASSEMBLER

CAPÍTULO 4

REFERÊNCIAS

CAPÍTULO 8

PLUS

CAPÍTULO 9

Apresentando o dBASE II

A partir de agora, usuários de micros padrão MSX podem finalmente utilizar o mais poderoso gerenciador de banco de dados criado para microcomputadores de 8 bits: o dBASE II, que amplia imensamente as possibilidades de aplicação da máquina.
Neste manual, você encontrará todas as funções e comandos do dBASE II agrupados em ordem alfabética. Cada comando ou função está acompanhado de explicações minuciosas sobre seu funcionamento e sintaxe.
O padrão MSX, de alta qualidade e baixo preço, foi criado pela Microsoft e por grandes empresas japonesas de hardware.
O dBASE é:
• um software notável: permite criar arquivos em disquetes e buscar dados com uma facilidade surpreendente. É possível, digamos, criar uma lista de fornecedores de uma loja de departamentos e depois pesquisar, por exemplo, quais são as malharias de São Paulo que fornecem mensalmente determinada quantidade de camisetas.
Tanto armazenar quanto recuperar dados são tarefas que podem ser feitas com comandos diretos ou através de programas, os quais o dBASE permite criar numa linguagem de programação de altíssimo nível. E para emitir relatórios, o usuário ainda tem comandos específicos. Assim, quando for necessário emitir listas de dados dos arquivos, será muito fácil obtê-las, e elas poderão ter cabeçalhos, numeração e organização do modo que o usuário escolher.
Usar um dBASE num MSX é muito fácil:
Basta ter um ou dois drives conectados ao micro. Caso estejam sendo usados drives da Microsol, Sharp ou da Gradiente deve-se carregar o dBASE com o sistema operacional MSX-DOS ou HB-DOS. Uma outra possibilidade é o uso do micro Hot Bit com drive Sharp, situação em que pode-se usar o sistema operacional HB-MCP, que é totalmente compatível com o sistema CP/M e permite uma velocidade de execução maior ao dBASE. Neste último caso é preciso fazer conversões de HB-DOS para HB-MCP, que dão um pouco mais de trabalho.
No disco existem duas versões de dBASE: o dBASE 4 e o dBASE 8.
Caso você tenha um cartão de vídeo de 80 colunas, poderá usar o dBASE 8 para trabalhar com uma tela de 80 colunas. Caso contrário, terá de usar mesmo o dBASE 4, que usa uma tela de 40 colunas.
Assim que o dBASE for carregado, aparecerá um pontinho, o **prompt**, indicando que o software está pronto para ser usado. Desse momento em diante você poderá abrir os arquivos que já houver montado, criar estruturas de arquivo (com tipos de dados: Alfanuméricos, Numéricos, Lógicos), dar comandos diretos de acesso aos arquivos, usar ou criar programas. E além dissso tudo, há mais um detalhe surpreendente: os arquivos criados com os sistemas operacionais MSX-DOS ou HB-DOS (que são totalmente compatíveis) podem ser lidos sem dificuldade alguma por um IBM-PC que esteja usando sistema operacional MS-DOS, pois, a formatação dos disquetes é absolutamente idêntica. E mais: programas escritos em dBASE para o MSX, podem rodar num IBM-PC.
Sem dúvida, a DATALOGICA está abrindo as portas do trabalho profissional para os micros brasileiros padrão MSX.

Requisitos do Sistema

O dBASE II REQUER AS SEGUINTES CARACTERÍSTICAS DE HARDWARE E SOFTWARE

• Microprocessador baseado em 8080, 8085 ou Z-80 (como TRS-80/II, Northstar, Apple II com cartão Z-80, MSX com DOS ou MCP (HB-MCP ou equivalente), etc.) com sistemas de operação CP/M Z.X, CDOS ou CROMIX.

• Um ou mais armazenadores externos (geralmente floppy disk drives)

• Cursor endereçável CRT se necessárias operações em tela cheia.

• Impressora – opcional (para alguns comandos) – com no mínimo 80 colunas.

NOTA

Alguns comandos não foram implementados para o MSX, pois apresentam resultados diversos quando usados com o cartão de 80 colunas, pois esse cartão ainda não é compatível com o MSX.

* MSX, HOT BIT, HB-DOS,HB-MCP, MS-DOS, TRS-80, APPLE II e CP/M são marcas registradas Microsoft, Gradiente, Sharp/Epcon, IBM,Tandy, Northstar e Apple respectivamente.

dBASE II
ESPECIFICAÇÕES

• números de campos por registro	32 max.
• número de caracteres por registro	1000 max.
• número de registro por banco de dados	65.535
• número de caracteres por string de caracteres	254 max
• precisão de campos numéricos	16 dígitos
• maior número	$1.8x10^{63}$ aprox.
• menor número	$1.0x10^{-63}$ aprox.
• número de variáveis de memória	64 max.
• número de caracteres por linha de comando	254 max.
• número de expressões em comando SUM	5 max.
• número de caracteres de cabeçalho REPORT	254 max.
• número de campos em REPORT	24 max.
• número de caracteres numa chave indexada	100 max.
• número de GETS pendentes	64 max.
• número de arquivos abertos ao mesmo tempo	15 max.
• tamanho de arquivo de comandos executável	ilimitado

dBASE II
Guia de Referência Rápida

NOTA INTRODUTÓRIA

O dBASE II - Guia de Comandos e Referências foi feito para suplementar o ARQUIVO HELP, dando acesso imediato a informações vitais para a operação do programa dBase II.

CONVENÇÕES TIPOGRÁFICAS

minúsculas	informação dada pelo usuário
MAIÚSCULAS	comando dBase II
[ ]	parte opcional de um comando
< >	informação dada pelo usuário

REGRAS PARA NOMES DE ARQUIVOS

O nome de um arquivo pode ter até 8 caracteres. Deve começar com uma letra, e pode conter letras, números e símbolos. Espaços em branco não são permitidos.

REGRAS PARA NOMES DE CAMPOS E VARIÁVEIS DE MEMÓRIA

Os nomes de campos, e de variáveis de memória podem ter até 10 caracteres. Devem começar com uma letra, e podem conter letras, números e 2 pontos (:). Espaços em branco não são permitidos.

TIPOS DE DADOS DO dBASE II

C - Dado tipo caracter. Pode conter até 254 caracteres de texto padrão.

L - Dado lógico, isto é, verdadeiro (.T./t/Y/y) ou falso (.F/.f./N/n).

N - Dado numérico para cálculos. Pode conter números, um sinal matemático e um ponto decimal.

CONTROLE DO CURSOR

As teclas de controle usadas nas operações de tela do MSX são:

TECLAS		FUNÇÃO
<seta para cima>	^E	Move o cursor uma linha para cima.
<seta para baixo>	^X	Move o cursor uma linha ou campo para baixo.
<seta à esquerda>	^S	Move o cursor um espaço para a esquerda. Em menus, move uma opção à esquerda.
<seta à direita>	^D	Move o cursor um espaço para a direita. Em menus, move uma opção à direita.
	^B	Move lateralmente um campo para a direita no BROWSE.
	^G	Apaga o caracter onde está o cursor.
	^F	Move o cursor uma palavra para direita.
	^W	Saída em Zoom. Sai e salva em operações de tela.
	^Q	Sai sem salvar as modificações (salva todos menos o registro atual no APPEND e no BROWSE) e retorna para o prompt.

NOTA
O símbolo (^) significa a tecla CTRL (Control).

TECLAS		FUNÇÃO
	^V	Liga e desliga o modo de inserção. Quando está ligado, insere o caracter antes do cursor. Quando está desligado, substitui o caracter onde está o cursor.
	^N	Insere uma nova linha ou definição de campo.
	^R	Retrocede um registro, uma tela, ou uma janela de 17 registros no BROWSE.
	^C	Avança um registro, uma tela, ou uma janela de 17 registros BROWSE.
<RETURN>		Move o cursor para o próximo campo ou linha. No APPEND, salva e sai se está no primeiro caracter de um registro em branco. No EDIT, salva e sai se está no último campo do registro.

Caracteres de Controle usados quando as operações de tela não estão em uso:

^H	Apaga enquanto retrocede.
^M	Tem o mesmo efeito da tecla **<RETURN>**

TERMINOLOGIA

<arquivo>	Nome do arquivo
<arquivo-índice>	Nome do arquivo-índice
<campo>	Nome do campo
<condição>	Expressão lógica
<chave>	Parte(s) de um arquivo usada(s) para criar o arquivo-índice.
<cstring>	Um agrupamento de caracteres
<dec>	Número de casas decimais
<delimitador>	Aspas simples, duplas, ou colchetes
<escopo>	Pode ser ALL, NEXT <N> e RECORD NUMBER
<exp>	Campos, variáveis de memória, funções ou combinações dos mesmos
<exp C>	Expressão tipo caracter
<exp N>	Expressão tipo numérica
<lista de campos>	Lista de nomes de campos separados por .AND. ou .OR.
<lista de expr>	Lista de expressões separadas por vírgulas
<n>	Um número
<mascára>	Coringas * e ? que substituem partes de um nome

OPERADORES

Aritméticos (em ordem de prioridade):

( )	parênteses para agrupar
*	multiplicação
/	divisão
+	adição
-	subtração

Relacionais:

<	menor que
>	maior que
=	igual a
< > ou #	diferente de
< =	menor ou igual
> =	maior ou igual

Lógicos (em ordem de prioridade):

.NOT.	NÃO lógico
.AND.	E lógico
.OR.	OU lógico

De string:

+	concatenação de strings
$	pesquisa de sub-string

Se a mesma expressão usar vários tipos de operadores, a prioridade é para os operadores 1) matemáticos, 2) relacionais e 3) lógicos.

As operações com o mesmo nível de prioridade são executadas da esquerda para a direita. Usam-se parênteses para alterar a prioridade.

INICIANDO

Se você tem um disco rígido, simplesmente copie todos os arquivos do disco de Exemplos de Arquivos e Programas, para um arquivo do disco rígido. Faça deste o diretório ativo no disco rígido; em seguida mude para o drive A. Insira no drive A o disco Sistema dBASE II ou disco de demonstração.

Com 2 disquetes, ative o drive A como drive default. Insira o disco de Sistema ou de Demonstração no drive A. Insira o disco de Exemplos de Arquivos e Programas no drive B.
Com disquetes ou disco rígido, sua tela exibirá este prompt:
`A>` quando o drive A é o drive ativo.
Você inicia o dBASE II do mesmo modo em disco rígido ou disquete.
Digite:

```
A>dBASE <return>
```

Será solicitada a data no formato DD/MM/AA.

> **NOTA**
> Se for informado o ano com 4 dígitos, como por exemplo, 1986, o ano será armazenado incorretamente.

O dBASE II responde com um sinal de identificação e uma nota da copyright e em seguida apresenta o "prompt" (sinal que indica que o sistema está pronto para receber seus comandos). O prompt de dBASE II é um ponto, e o cursor apresenta-se logo vizinho a este.

O ponto indica que o dBASE II está no modo interativo e pronto para seus comandos. No modo interativo, sempre que entrar com um comando, você obtém uma resposta imediata. Mostraremos os comandos com letras maiúsculas, mas você pode usar maiúsculas ou minúsculas.

Você deverá avisar ao dBASE II onde estão os arquivos de dados.
Digite:

`SET DEFAULT TO B: <return>` disquetes

Você pode finalizar uma sessão de dBASE II quando quiser, digitando:

```
.QUIT
```

Vamos agora aprender a usar o dBASE II.

LINGUAGEM DE MÁQUINA

Comandos de Linguagem de Máquina

SET CALL TO <ENDERECO>

Arranje o endereço decimal que será chamado pelo comando CALL DO dBASE.

CALL [<**VARMEM**>]

Execute uma chamada de linguagem de máquina para o endereço colocado por um SET CALL TO ou o endereço padrão se nenhum SET CALL tiver sido feito. Há cerca de 254 bytes de stack disponíveis; o par de registros HL indica o primeiro byte se a <**varmem**> for uma string.
É muito importante que nenhuma tentativa seja feita para o alocamento ou endereçamento da string.
O Controle pode ser devolvido ao dBASE com a instrução RET.

LOAD [<**ARQUIVO**>]

Este comando carrega um arquivo que assumi ser um arquivo .HEX no formato INTEL HEX na memória.

POKE <**ENDERECO**>, <**DADOS BYTE**> [,<**DADOS BYTE**>...]

Digite os dados diretamente na memória.

PEEK (<**endereco**>)

Uma função que retorna um número correspondendo ao valor binário não endereçado do byte no <**endereco**>.

> NOTA
> Cuidado com os pokes aleatórios para não serem destruídas áreas do programa dBASE ou variáveis.

ARQUIVOS DO dBASE II

O dBase II armazena informações como arquivos em disco em sete formatos especiais. Cada um serve para uma necessidade específica de processamento do dBASE II.

Cada arquivo em disco tem um nome de no máximo 8 caracteres e um identificador que consiste de um ponto e 3 caracteres.

O nome do arquivo é destinado pelo usuário no momento em que o arquivo é criado. O identificador deve ser designado ao mesmo tempo. Normalmente o dBASE II designa o identificador de acôrdo com o tipo de arquivo sendo criado. Contudo, o usuário pode designar outro identificador, simplesmente digitando uma extensão à sua escolha no momento de entrar com o nome do arquivo designado pelo dBASE II é o default. Estes identificadores, assim como os sete tipos de arquivos, são definidos na seguinte tabela:

Tipo de arquivo	Extensão de arquivo
Arquivo de Dados	.dbf
Índice	.ndx
Arquivo de memórias	.mem
Comandos	.cmd
Arquivo de formatação de tela	.fmt
Arquivo de formato de relatório	.frm
Saída de texto	.txt

Se um outro codificador que não o default for usado, deve ser incluído com o nome do arquivo sempre que este é selecionado para uso.

ARQUIVOS DE BANCO DE DADOS (.DBF)

Armazenam dados em registros e campos (linhas e colunas). Cada registro se destina a conter um conjunto único de informações.

Os arquivos de banco de dados do dBASE II podem englobar até 65.535 de registros. Cada registro pode conter até 1000 bytes, e não deve conter mais que 32 campos de dados.

ARQUIVOS DE ÍNDICE (.ndx)

Os arquivos de Índice possibilitam usar um banco de dados numa ordem lógica e não física. A ordem física é aquela na qual os registros foram introduzidos. A ordem lógica é uma ordem alfabética ou numérica baseada no conteúdo de um ou mais campos de banco de dados.

Arquivos de Índice fazem a correspondência entre uma chave (um item de interesse como por exemplo um nome) e o número do registro correspondente do banco de dados. Quando usamos um banco de dados com um arquivo índice, o banco de dados aparentará estar na ordem do item chave. Isto se chama ordem lógica. A chave (que pode ser um ou mais campos) pode também ser usada para fornecer acesso direto (randômico) a um determinado registro.

ARQUIVOS DE COMANDO (.cmd)

Os arquivos de comando contém conjuntos de instruções do dBASE II armazenadas como programas. São arquivos padrão ASCII e podem ser criado com MODIFY COMMAND ou com o modo não-documento da maioria dos processadores de textos.

ARQUIVOS DE FORMATAÇÃO (.fmt)

Criam formatos de tela especificados pelo usuário para serem usados na entrada de dados e saídas em impressora.

ARQUIVOS DE MEMÓRIA (.mem)

Contém até 64 variáveis de memórias. São usados para salvar conteúdo das variáveis para uso posterior. Estes arquivos são criados pelo comando SAVE e restaurados pelo comando RESTORE.

ARQUIVOS DE FORMATO DE RELATÓRIO (.frm)

Contém informações necessárias para o comando REPORT preparar relatórios. Podem ser criados e alterados pelo comando MODIFY.

ARQUIVOS DE SAÍDA DE TEXTO (.txt)

Arquivos de saída de um texto são usados basicamente como interface entre o dBASE II e outros softwares. Esses arquivos são padrão ASCII e contém somente os caracteres ASCII imprimíveis. Os arquivos de texto serão lidos em um arquivo de dados dBASE II por uma forma especial do comando APPEND FORM. São criados por uma forma especial do comando COPY. Também podem ser usados para registrar operações de processamento do dBASE II por meio do comando SET.

ESTRUTURA DO BANCO DE DADOS

A estrutura ou máscara de um arquivo de dados é estabelecida definindo-se cada um dos campos de dados. Faz-se com os comandos CREATE e MODIFY STRUCTURE. A definição de campo consiste de:

NOME DO CAMPO

Pode ser até 10 caracteres. Deve começar com uma letra e não pode conter espaços em branco. Letras, números e sublinhados são permitidos.

TIPO DO CAMPO

Os tipos de campos usados no dBASE II são:

C - Caracter
L - Lógico
N - Número

Estes tipos de campo vêm descritos na página seguinte.

TAMANHO DO CAMPO

Tamanho de campo é o número máximo de caracteres que podem estar contidos num campo. No caso de campos numéricos o ponto (se houver) é contido como um dígito.

As características dos três tipos de variáveis são:

Caracter – Normalmente usada para armazenar strings de caracter. Entretanto, uma sequência binária pode ser armazenada numa string através da função CHR. As variáveis caracter podem conter até 254 caracteres. O espaço total de memória usado por uma variável de caracter é seu tamanho em bytes mais dois bytes adicionais.

Numérica – Usada para armazenar números que podem ser usados em cálculos. A precisão das variáveis numéricas estendem-se a 16 dígitos.

Lógica – Variáveis lógicas são armazenadas como Verdadeira (.T.) ou Falsa (.F.). O tamanho de uma variável lógica é sempre um byte. Ela requer um total de 2 bytes de memória. Variáveis lógicas aceitarão T, t, Y ou y para True e F, f, N ou n para False. Quando criadas a partir do teclado, o valor lógico deve ser delimitado por pontos.

CAMPOS

Há três tipos de campos usados no dBASE II. São eles:

Campos caracter – Podem ser usados para armazenar qualquer caracter imprimível que possa ser entrado via teclado (incluindo espaços em branco). Estes são os caracteres ASCII imprimíveis que incluem: letras, números, símbolos especiais e espaços. O tamanho de um campo caracter é no máximo 254.

Campos numéricos – São de dois tipos: inteiro e decimal. Número inteiro é aquele que não tem parte decimal (por exemplo, o número de alunos numa sala de aula). O tamanho de um campo numérico é o número de dígitos, que o campo pode conter (o ponto decimal conta como um dígito). A precisão de um campo numérico se estende 16 dígitos.

Campos lógicos – Aceitam somente um caracter representando valores verdadeiro/falso (True/False). True lógico (.T.) é registrado como T, t, Y ou y. False lógico (.F.) é registrado como F, f, N ou n.

VARIÁVEIS DE MEMÓRIA

Variáveis de memória são dados que têm um nome e que são armazenados independentemente, fora de estrutura do banco de dados. Elas proporcionam um conveniente meio de armazenamento temporário quando criadas via teclado. Por exemplo, o conteúdo de um campo pode ser somado e o resultado armazenado como variável de memória. O conteúdo desta então será diretamente usado em cálculos posteriores. Elas são indispensáveis em aplicações programadas onde proporcionam os meios para controlar o programa.

As variáveis de memória são semelhantes aos campos de dados, e podem substituí-los em muitas aplicações (sempre que a palavra "expressão" for usada na sintaxe do comando).

Há três tipos de variáveis de memória: caracter, numérica e lógica, pode haver até 64 variáveis de memória ativas as quais combinadas não podem exceder o tamanho de 1536 bytes.

Os nomes de variáveis de memória podem ter até 10 caracteres e podem conter letras e números. Devem começar com uma letra e não podem conter espaços em branco.

Não há restrição de nomes variáveis dè memória, entretanto, problemas podem surgir quando uma variável tem o mesmo nome que um campo no arquivo de dados. Os campos têm precedência sobre as variáveis de memória quando à ambiguidade. Pode-se eliminar a ambiguidade precedendo o nome da variável de memória por M.

As variáveis de memória são normalmente usadas para armazenagem temporária de dados. Entretanto elas podem ser salvas num arquivo em disco (.mem) usando o comando SAVE, e podem ser chamadas de volta de um arquivo de memória por meio de comando RESTORE.

Variáveis de memória podem ser criadas por qualquer um dos seguintes comandos: ACCEPT, COUNT, SUM, INPUT, STORE e WAIT.

Normalmente, altera-se o conteúdo de uma variável registrando-se um novo valor por cima do valor anterior, isto é, usando o mesmo nome da variável para obter um novo valor. O único comando que pode realmente editar o conteúdo de uma variável existente é @ GET.

REGRAS OPERACIONAIS

Quando usamos a linguagem de comando do dBASE II, há certas regras que devem ser seguidas para nos certificarmos que os comandos sejam escritos corretamente:

1. Cada comando deve começar com um verbo de comando existente na seção de Comandos deste manual de referência.

2. Cada comando deve seguir a respectiva sintaxe como descrito na seção de Comandos deste manual de referência.

3. Um comando consiste de um verbo e opcionalmente uma ou mais cláusulas que qualificam o verbo. Cláusulas opcionais (por exemplo, as que começam com FOR. WHILE e NEXT) podem ocorrer em qualquer ordem, por exemplo:

```
DISPLAY NEXT 25 FOR NOME = "Jaime"
DISPLAY FOR NOME = "Jaime"  NEXT 25
```

4. O tamanho máximo de um comando é 254 caracteres.

5. Palavras dentro de um comando podem ser separadas por qualquer número de espaços em branco. Estes espaços em branco, entretanto, fazem parte do limite de 254 caracteres.

6. Comandos e certas palavras-chave podem ser abreviadas para os primeiros quatro caracteres (por exemplo, DISPLAY MEMORY pode ser abreviado como DISP MEMO).

7. Verbos de comando, palavras-chave, nome de campo, nomes variáveis de memória e nomes de arquivos podem ser escritos em, letras maiúsculas ou minúsculas ou em qualquer combinação delas.

8. Embora o dBASE II não tenha palavras reservadas, é aconselhável evitar o uso de palavras-chave para nomes de arquivos de campos ou variáveis de memória, para evitar dificuldades. Por exemplo, um campo chamado Status não poderia ser mostrado pelo comando DISPLAY STATUS, porque o comando mostraria o Status corrente do processamento, em vez do conteúdo do campo Status.

9. Quando programando, tome cuidado para que todas as declarações DO WHILE/ENDDO, DO CASE/ENDCASE e IF/ENDIF, estejam inteiramente dentro uma da outra. Se por exemplo uma declaração IF é usada dentro de um LOOP, o IF e seu respectivo ENDIF devem estar contidos entre o DO WHILE e o ENDDO.

```
DO WHILE.NOT. EOF()
 IF TRUE
 ? DATALOGICA
 ENDIF
ENDDO
```

SINTAXE

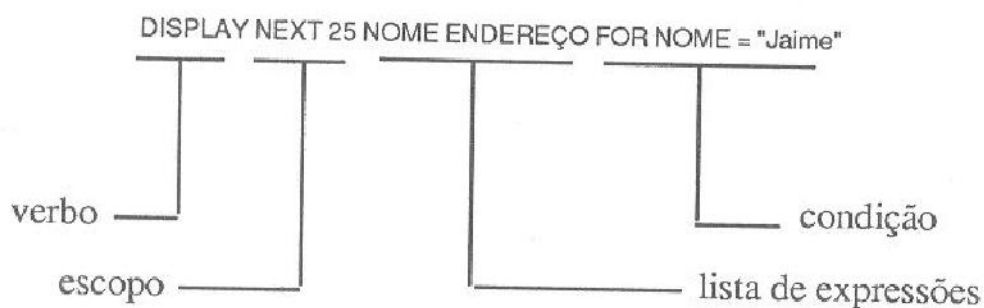
A estrutura de um comando é sua sintaxe. Cada comando começa com um verbo que é o comando básico. Muitos comandos também têm uma ou mais cláusulas que foram feitas sob medida para o comando a fim de atender a uma necessidade particular. A sintaxe geral de um comando é ilustrada por:

VERBO [<escopo>] [<lista de expressões>] [FOR/WHILE <condição>].

A sintaxe específica para cada comando é mostrada na seção de Comandos deste manual.

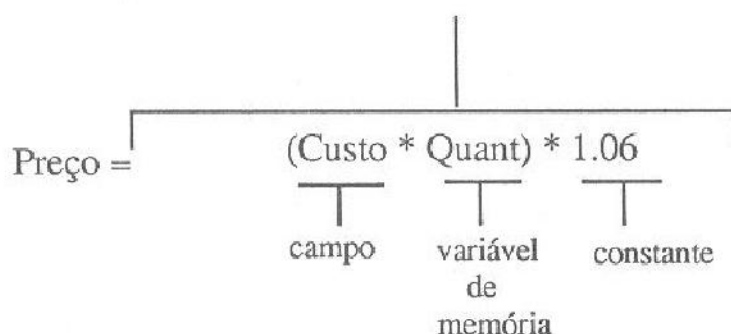
Os itens incluídos entre colchetes são opcionais, e podem ser entrados em qualquer ordem. Itens em letra maiúscula são registrados exatamente como mostrado na seção de Comandos. Itens incluídos entre colchetes angulares (< >) são fornecidos pelo usuário. Os termos usados na sintaxe geral são descritos na seção Palavra-Chave deste manual.

Não digite os colchetes retos ([ ]) ou angulares (< >) quando entrar com um comando.

Ilustrando a sintaxe de um comando típico:

EXPRESSÕES

O termo "expressão" é usado de várias formas, em toda esta seção de Referência,. Usa-se "expressão" porque é mais geral do que campo, e porque o dBASE II pode operar com expressões, que são mais complexas que um campo. Uma expressão pode ser formada por qualquer combinação de:

Campos
Variáveis de Memória
Constantes
Funções
Operadores

Ilustrando: Um banco de dados contém o campo numérico Custo. Uma variável de memória numérica Quant contém a quantidade vendida. A taxa de imposto local é de 60%. Isto significa que o preço para o consumidor é:

(CUSTO * QUANT) * 1.06 é uma expressão numérica. Nessa seção de Referência, ela pode ser indicada por expressão numérica, expressão, exp, ou expN - dependendo do comando ou função, e do espaço disponível.

OPERADORES

O dBASE II tem quatro tipos de operadores:
matemáticos, relacionais, lógicos e de string.

OPERADORES MATEMÁTICOS

Geram resultados numéricos.

+	= adição
-	= subtração
*	= multiplicação
/	= divisão
( )	= Parênteses para agrupar

OPERADORES RELACIONAIS

Operadores de comparação geram resultados lógicos (por exemplo, True ou False). Podem ser usados com variáveis caracter ou numéricas. Ambas as expressões devem ser do mesmo tipo.

<	= menor que
>	= maior que
=	= igual a
< > ou #	= diferente
< =	= menor ou igual
> =	= maior ou igual

OPERADORES LÓGICOS

Obtêm um resultado lógico da comparação de duas expressões lógicas.

.AND.	= E lógico
.OR.	= OU lógico
.NOT.	= NÃO lógico (trabalha com uma expressão única)

OPERADORES DE STRING

+	= Operador de concatenação. É usado para juntar dois ou mais strings de caracter em uma única string de caracter.
$	= Substring de comparação (por exemplo, se A e B são strings de caracter, A$B retorna um valor lógico True se A for idêntico a B ou se estiver contido em B).

PRECEDÊNCIA DOS OPERADORES

Cada tipo de operador tem um conjunto de regras que governam a ordem na qual as operações são executadas, e que se chamam níveis de precedência do operador.

A comparação e operadores string tem somente um nível de precedência. Os operadores dessa categoria são executados em ordem sequencial da esquerda para a direita.

Os níveis de precedência para operadores matemáticos são:

1. Sinais + e -
2. Multiplicação e divisão
3. Adição e Subtração

Os níveis de precedência para os operadores lógicos são:

1. .NOT.
2. .AND.
3. .OR.

Quando vários dos quatro tipos de operadores são usados na mesma expressão, os níveis de precedência para os tipos são:

1. Matemático e de string
2. Comparação
3. Lógico

Todas as operações no mesmo nível de precedência são executadas da esquerda para a direita. Use parênteses para alterar a ordem na qual as operações são executadas.

FORMATOS DE TELAS ESPECIAIS

Pode-se criar e usar formatação de tela especial, definida pelo usuário, para entrar com dados e mostrá-los no vídeo.

Esses formatos de telas são salvos em disco como arquivos de formato (.FMT). Um exemplo de formato especial é mostrado abaixo:

```
                              Dados Essenciais              04/25/84
     ==============================================

     NOME           [Marcos   ]
     IDADE          [37  ]
     ENDERECO    [R. Cavitar, 782  ]
     CIDADE        [Blumenau  ]     ESTADO [SC]   CEP [24000]
```

Um arquivco de formato é um tipo especial de programa que contém somente comandos @...SAY, @...GET. Um exemplo de um arquivo de formato é mostrado abaixo. Esse exemplo cria o formato especial acima, quando usado em conjunção com o arquivo de dados Exemplo e a curta sequência de comandos mostrada logo abaixo do arquivo de formato. Colchetes ([ ]) são usados para destacar os campos no exemplo ao invés do vídeo reverso normal. Isto foi realizado usando o comando SET COLON:

```
@ 5,22 SAY "Dados Essenciais"
@ 5,22 ? DATE()
@ 6,1 SAY "===================="
@ 8,10 SAY "NOME" GET MNOME
@ 9,10 SAY "IDADE" GET MIDADE
@ 10,10 SAY "ENDERECO" GET MENDERECO
@ 11,10 SAY "CIDADE" GET MCIDADE
@ 12,10 SAY "ESTADO" GET MESTADO
@ 13,10 SAY "CEP" GET MCEP
```

e é mostrado pelos seguintes comandos do dBASE II:

```
USE Cadastro
SET FORMAT TO Dados
SET COLON TO "[]"
SET COLON ON
EDIT
```

Um arquivo de formato pode ser criado usando-se o processador de texto interno MODIFY COMMAND.
Um arquivo de formato é aberto pelo comando SET FORMAT TO. Sempre que os comandos APPEND, CHANGE, EDIT, ou INSERT são usados, o formato especial irá substituir o formato padrão usado por esses comandos. O comando READ também ativa o formato especial.
Para cada arquivo de dados aberto, pode haver um arquivo de formatação separado. Entretanto, o número total de arquivos abertos não deve ultrapassar 15.

INDEXAÇÃO

dBASE II proporciona um método de usar os registros de dados numa ordem lógica, sem rearranjá-los fisicamente. Este processo de ordenação de registros é conhecido como INDEX.

O INDEX cria um arquivo auxiliar que é usado em conjunção com um arquivo de dados. Este arquivo índice (.ndx) fará os registros aparecerem numa ordem específica.

A especificação da ordem desejada é realizada pelo uso de uma chave, que pode ser qualquer item de interesse contido num registro. A chave deve ser um caracter, data, ou expressão numérica válida. O uso da chave fará com que os registros sejam apresentados alfabeticamente, cronologicamente ou numericamente.

Para ilustrar, o banco de dados Alunos é indexado pelo campo Nome. Os registros aparecem em ordem alfabética. Observe que os números dos registros estão fora de sequência. O número do registro indica a posição física de um registro no arquivo de dados. A ordem indexada é a ordem lógica dos registros (para esta chave).

NUM. DO REGISTRO	NOME	IDADE	CIDADE
297	FABIO	40	SANTOS
125	GERALDO	29	R. DE JANEIRO
1	JULIO	20	B. HORIZONTE
714	LAURINDO	35	MACAÉ
27	WAGNER	27	VILA VELHA

O arquivo índice é usado para achar um determinado registro rapidamente, quando se pesquisa por uma chave. A estrutura do arquivo índice é projetada para ser pesquisada rapidamente. Para arquivos não muito pequenos, o tempo de procura médio é drasticamente reduzido quando se usa um arquivo de índice.

Pode-se ter um número qualquer de arquivos de índice associados a um arquivo de dados. Até sete arquivos índice podem estar abertos ao mesmo tempo. A ordem lógica do arquivo de dados é determinada pelo primeiro arquivo índice aberto. Para mudar o arquivo-índice que controla os dados, fechamos todos os arquivos-índice abertos e em seguida os reabrimos na ordem desejada. Todos os arquivos de índice abertos são atualizados quando o conteúdo do banco de dados sofre alterações.

Uma chave de índice pode ter até 100 caracteres de tamanho. O tempo que se leva para criar e atualizar um arquivo índice cresce com o tamanho do campo chave.

Os comandos principais associados com indexação são: INDEX, SET INDEX TO, FIND, REINDEX, SET LINKAGE e USE.

TIRANDO BACKUPS
(CÓPIAS DE RESERVA)

Todos os arquivos devem ser copiados sistemática e regularmente, para proteger contra perdas. Pode-se fazer cópias backups através do sistema operacional ou pelo próprio dBASE II. Para usar os utilitários de cópia e backup do sistema operacional, consulte o manual do mesmo. Para fazer cópias através do dBASE II, use o comando COPY.

Somente arquivos .DBF podem ser copiados dentro do dBASAE II. Caracteres coringa, tais como * e ? não são permitidos no comando COPY. Outros tipos de arquivos deverão ser copiados pelos comandos COPY ou PIP de seu sistema operacional.

Para copiar um arquivos DBF do drive default (corrente) para o drive B:

```
COPY Exemplo .DBF TO B: Exemplo .dbf
```

NOTA
Dados novos ficam desprotegidos e em risco até serem copiados nos backups.
Para cópias do sistema DOS para MCP deve-se utilizar o programa COPDOS que se encontra no disco do MCP.

PALAVRAS-CHAVE E SÍMBOLOS

Essa seção do manual cobre as palavras e símbolos que tem um significado especial quando usadas para descrever os comandos e funções do dBASE II.

< > Colchetes angulares indicam que o item deve ser fornecido pelo usuário. Os colchetes não fazem parte do comando.

[ ] Colchetes retos indicam que o item pode ser omitido, é opcional. Os colchetes não fazem parte do comando.

^ Indica a tecla de controle. Quando esse símbolo é usado, a tecla de controle <**control**> é pressionada e mantida assim enquanto uma outra tecla é pressionada. Por exemplo, ^U significa que a tecla de controle deve ser mantida pressionada para baixo enquanto a tecla de letra U é pressionada.

/ A barra indica uma escolha entre opções.

Arquivo Sempre se refere a um arquivo em disco. Dentro de um comando, ele é seguido por um nome de arquivo para indicar que o comando deve ser aplicado ao arquivo inteiro (por exemplo DELETE FILE parcial).

Nome do Arquivo O nome de um arquivo em disco. Um nome de arquivo pode ter 8 caracteres, sendo constituído de letras, números e sublinhados. Deve iniciar com uma letra e não pode conter espaços em branco.

Arquivo Ativo O arquivo e banco de dados usado na área de trabalho atual.

Coluna Um número usado para indicar a coluna inicial no monitor de vídeo ou na impressora. Valores da coluna crescem da esquerda para a direita, com o menor valor sendo igual a zero. Na impressora, a posição da coluna é somada ao valor da margem esquerda (veja SET MARGIN).

Comando Uma instrução do dBASE II para o computador. Estão descritos na seção de Comandos deste manual.

Condição Define mais especificamente o que um comando deve fazer. A condição por si só é uma comparação entre 2 ou mais itens. Por exemplo, no comando: DISPLAY FOR NOME = "silva". O conteúdo do campo Nome é comparado com a string de caracter "silva" para cada registro do arquivo de dados, somente os registros nos quais esta condição seja verdadeira serão mostrados. Condições múltiplas podem ser definidas usando-se os lógicos AND e OR.

String Uma abreviação de string de caracter.

Default Uma escolha pré-programada. Esta é a ação ou seleção que o dBASE II vai executar até que o usuário designe algo em contrário.

Delimitadores Usados para definir ou identificar separadamente os dados para o computador. Dados do tipo caracter normalmente são indicados por aspas simples ou duplas (por exemplo, STORE [OLA] TO X). O mesmo deve ser usado em ambas as extremidades da string de caracter. Dados do tipo caracter também podem ser separados por colchetes (como STORE 'OLA' TO X). Vírgulas são usadas para separar dados numa lista (por exemplo DISPLAY Nome, endereço, Estado).

Drive Indica o identificador de drive (por exemplo, A ou B). Quando usado como parte do nome de um arquivo, o identificador de drive deve ser seguido por dois pontos (por exemplo, A:)

Escopo Parte opcional de muitos comandos, que especifica a extensão do banco de dados a que o comando se aplica. Cada comando que tem opção escopo, tem um default de registro decorrente do banco de dados, ou de todos os registros. As expressões FOR/WHILE colocam o default do escopo para ALL. O escopo pode ser:

RECORD n – um único registro
NEXT n – n registros começando com o registro corrente
ALL – todos os registros do arquivo de dados.

Exp Abreviação de expressão.

Expressão Pode consistir de um campo, uma variável de memória, uma função, uma constante, ou qualquer combinação válida destes. Expressões podem ser designadas pela abreviação para indicar o tipo de dado da expressão (por exemplo, <expC> significa uma expressão de caracter).

Lista de Expressões Consiste de uma ou mais expressões. Se há mais de uma expressão, devem ser separadas por vírgulas.

Campo Contém um item de informação indefinida num arquivo de dados.

Linha Um número usado para indicar a linha tanto do monitor de vídeo como da página impressa. As linhas são numeradas de cima para baixo, começando pela linha 0.

Lista de Campos Consiste de um ou mais nomes de campos. Quando há mais de um nome de campo, os nomes devem ser separados por vírgulas.

Nome do Campo O nome designado para um campo. Um nome de campo pode ter até 10 caracteres, consistindo de letras, números e sublinhados. Deve começar com uma letra e não pode conter espaços em branco.

Tipo de Campo De acordo com o conteúdo: caracter, lógico ou numérico.

FOR <condição> Uma cláusula opcional em muitos comandos, que indica que o comando deve ser aplicado a cada registro de dados em que a condição for verdadeira.

Chave Veja Campo Chave.

Campo Chave	Um campo ou expressão pela qual um arquivo de dados é indexado (INDEX), agrupado (JOIN), atualizado (UPDATE) ou totalizado (TOTAL).
Máscara	Estrutura ou gabarito usado para definir nomes de arquivos e nomes de variáveis de memória que tenham elementos em comum. O esqueleto usa o ponto de interrogação (?) e (*) para partes coringas desses nomes com elementos comuns. O * se aplica a vários caracteres. O caracter ? é um coringa para um único caracter. Para ilustrar, list file like ??SE. * mostrará os nomes dos arquivos onde o nome é de 4 caracteres de comprimento, terminando com SE e a extensão do arquivo pode ser qualquer uma.
Varmem	Abreviação de variável de memória.
Lista de Varmem	Consiste de um ou mais nomes de variáveis de memória. Se há mais de uma, os nomes devem ser separados por vírgulas.
n	Indica um número (por ex.: 2137). Na maioria dos casos, pode-se usar também uma expressão numérica (por ex.: custo * percentagem).
Parâmetro	Refere-se a: as características de processamento ajustáveis do dBASE II que são controladas pelos comandos SET. Por exemplo, o estado da impressora (ON ou 0FF).
Print	Indica a saída de um comando para a impressora. TO PRINT é uma parte opcional de certos comandos que ligam a impressora apenas enquanto o comando está ativo. A impressora também pode ser ligada ou desligada por meio do comando SET.

Strings de Caracteres

Uma sequência de letras, números e símbolos. A sequência Av Paulista 2020, cj 54, São Paulo, SP é uma string de caracteres. Os conteúdos dos campos caracter ou das variáveis de memória tipo caracter podem normalmente ser substituídas por strings de caracteres. Strings de caracteres vêm normalmente entre aspas ou outros delimitadores.

TO

Parte de muitos comandos do dBASE II que indica o alvo do comando. TO é normalmente seguido de um nome de arquivo ou de variável de memória. Alguns comandos usam TO como conector natural para especificar uma característica de processamento (por exemplo, SET alternate TO).

WHILE <condição>

Elemento opcional em muitos comandos em que o comando deve ser repetido enquanto a condição estabelecida for verdadeira. Quando o WHILE <condição> é usado com um comando que se aplica a registros do banco de dados (por ex.: LIST), cada registro é conferido sequencialmente, e o comando termina quando a condição se torna falsa, ou quando se ultrapassa o escopo.

WITH

Usado nos comandos onde o conteúdo de um campo é substituído por um novo valor. WITH é também usado para indicar os parâmetros a serem passados, usando o comando DO.

> **NOTA**
> Em vários comandos foi utilizado o arquivo Cadastro. Sua estrutura está na página 6-24 e a listagem de seu conteúdo na página 6-35 (Indexed) e 6-65 (Ordem Nome).

COMANDOS

?/??

Significa aproximadamente "O que é...?" ou "Qual o valor de...?"

Sintaxe:
?/?? <lista de expressões>

Observações:
Um único ponto de interrogação faz com que haja um retorno de carro (carriage return) e nova linha antes de mostrar a lista de expressões. O duplo sinal de interrogação não faz o mesmo. Quando usado em um programa, o ?? mostra a lista de expressões a partir da posição corrente do cursor, ou da cabeça de impressão. O ? sem ser seguido por uma expressão mostrará uma linha em branco.

Exemplos:
O arquivo de banco de dados Cadastro será usado para ilustrar o comando. Para mostrar o conteúdo do campo "NOME" e "CIDADE" do registro corrente.

```
.USE Cadastro
. ? NOME CIDADE
Marcelo Vianna Manaus
```

Para determinar se o NOME no registro corrente é Marcelo Vianna:

```
.? NOME = "Marcelo Vianna"
.F.
```

Para mostrar o resultado de um cálculo:

```
.? 2 * 4 + (9 / 3)
11
```

Para mostrar da data do sistema:

```
. ? DATE()
05/15/84
```

@

O comando @ é usado para criar um formato especial para entrada e saída de dados. Ele mostra informações numa formatação específica, através de um conjunto de coordenadas.

Sintaxe:
@ <linha, coluna> [SAY <exp> [USING <cláusula>]] [GET <variável> [PICTURE <cláusula>]]

Observações:
Linha e coluna são expressões numéricas. Para um terminal 24x80, a coordenada da linha pode ir de 0 a 23 e a coordenada da coluna de 0 a 79. A linha 0 (zero) é reservada como linha de status; assim, o usuário deve evitar usar 0 como coordenada de linha para operações de vídeo. Para a maioria das impressoras, também existem limitações físicas nas coordenadas linha e coluna. Diminuir o número da linha em consecutivos comandos @ vai causar um avanço de página. Diminuir o número de coluna pode ter efeito semelhante.
A opção SAY é usada para mostrar uma informação constante. A expressão deve ser uma expressão válida do dBASE II de qualquer tipo.
A opção GET é usada para mostrar uma informação variável, que pode ser uma variável de memória existente ou qualquer campo do arquivo de dados em uso. Esta parte do comando não chama o modo de tela, mas deve ser ativada por um comando de tela tal como READ ou APPEND.
@ linha, coluna apaga a linha a partir da coluna especificada.
A opção USING dá ao usuário uma saída formatada, e a PICTURE restringe os dados que podem ser aceitos por uma variável. A cláusula pode consistir de uma função e/ou uma máscara que vai ser descrito abaixo – e deve ser delimitada. Se uma função for usada, o símbolo @ deve aparecer como o primeiro caracter na cláusula. Se a função for usada em conjunto com uma máscara, um espaço deve separar os dois.

Símbolos de Máscaras

9 – Permite somente dígitos para dados tipo caracter; dígitos e sinais para dados numéricos.

– Permite somente dígitos, brancos e sinais.

A – Permite somente letras.

L – Permite somente dados lógicos.

X – Permite qualquer caracter.

! – Converte letras para maiúsculas e não afeta os outros caracteres.

$ – Mostra cifões no lugar de zeros à esquerda.

* – Mostra asteriscos ao invés de zeros à esquerda.

. – Especifica a posição decimal.

, – Vírgula somente será mostrada se houver números à esquerda.

O único símbolo que pode efetivamente ser usado em ambos os comandos SAY e GET é o "!".
Ele mostrará a informação em letras maiúsculas, sem levar em consideração como a informação foi digitada, e forçará a entrada em letras maiúsculas. Se a máscara PICTURE é usada para se obter com GET um número com parte decimal, o ponto decimal deve ser colocado na máscara.

ACCEPT

ACCEPT é usado basicamente em arquivos de comando para indicar ao usuário que faça uma entrada via teclado. A entrada pelo teclado é armazenada como string de caracter, sem necessidade do uso de delimitações. A entrada de dados é determinada por um <RETURN>.

Sintaxe:
ACCEPT <mensagem> TO <varmem>

Observações:
A variável pode ser tanto uma variável do tipo caracter ou uma string literal. Quando se usa uma literal, a mensagem deve ser delimitada por aspas simples, aspas duplas ou colchetes.

Se o operador entra com um <RETURN> em resposta ao comando ACCEPT, o conteúdo das variáveis de memória será nulo (ASCII 0).

O número máximo de caracteres que podem ser digitados em uma variável via ACCEPT é 64.

Exemplos:
Para indicar ao usuário que digite o endereço, e armazenar a entrada de teclado na variável de memória Endereço:

```
.ACCEPT "Digite o endereco "TO Endereco <return>
```

A tela mostrará:

```
Digite o endereco: <return>
```

Veja também: INPUT, WAIT

APPEND

APPEND acrescenta novos registros ao fim do arquivo de dados em uso. É um comando básico para a entrada de dados em modo de tela.

Sintaxe:
APPEND [BLANK]

Observações:
APPEND coloca o usuário no modo de tela para entrada de dados. Um registro por vez é apresentado em uma tela formatada para entrada de dados.

O processo é terminado por um <RETURN> quando um novo registro é apresentado.

Todos arquivos de índices ativos durante o APPEND serão atualizados, incluindo os novos registros.

O diretório do arquivo ativo pode deixar de listar os novos registros que vão sendo acrescentados. O diretório só será totalmente atualizado depois que o arquivo for fechado. Se a opção BLANK é usada, um registro em branco é adicionado ao fim do arquivo de dados e não se entra no modo novo registro inserido.

Exemplo:
Usaremos o arquivo Cadastro no exemplo abaixo. Para entrar no modo de tela para entrada de dados, e começar a acrescentar registros ao arquivo de dados:

```
.USE Cadastro <return>
.APPEND <return>

RECORD # 11
NOME:
IDADE:
ENDERECO:
CIDADE:
ESTADO:
CEP:
```

Veja também: MODIFY COMMAND, SET CARRY, SET FORMAT TO

APPEND FROM

APPEND FROM copia registro de um arquivo fonte para o arquivo de dados ativo. Os registros são sempre adicionados ao fim do arquivo receptor. O arquivo fonte não precisa ser um arquivo dBASE II. APPEND FROM é o comando básico para transferir dados de outras fontes de fora do dBASE II.

Sintaxe:
APPEND FROM <nome do arquivo> [FOR/WHILE <condição>] [SDF/DELIMITED]

Observações:
O nome do arquivo deve incluir a designação do drive, se não estiver no drive corrente.

Se a extensão do arquivo não for especificada como parte do nome do arquivo, e a opção SDF/DELIMITED não for escolhido, o dBASE II vai assumir a extensão .dbf.

Se a extensão de arquivo não for especificada como parte do nome do arquivo e a opção SDF/DELIMITED for escolhida, o dBASE II vai assumir a extensão .txt.

Se o arquivo fonte é um arquivo de dados dBASE II:

- Somente registros do arquivo fonte que não foram marcados para eliminação serão copiados.
- Somente campos que forem comuns aos dois arquivos serão copiados (nome e tipo do campo).
- Se o arquivo for maior que o arquivo receptor, os dados do tipo caracter serão truncados, mas asteriscos serão colocados para os dados numéricos.
- A expressão FOR/WHILE pode apenas referenciar nomes de campos que são comuns aos dois arquivos.

Se a opção SDF é usada, os dados serão copiados caracter por caracter, começando pela esquerda. O final de um registro SDF é indicado por um sinal de volta do carro (CR).

Se a opção DELIMITED é usada, os dados serão copiados campo a campo, começando pela esquerda. O fim de um campo é indicado por uma vírgula. O fim de registro delimitado é indicado por um sinal de volta do carro (CR).

Importante:
O ARQUIVO QUE ESTIVER EM OUTRA LINGUAGEM (BASIC, COBOL, FORTRAN, ETC) DEVERÁ ESTAR DESCOMPACTADO E EM LINE SEQUENCIAL PARA SER TRANSFERIDO PARA O dBASE II.

Exemplos:
Para anexar os registros do arquivo de dados Alunos para o banco de dados Classe:

```
.USE Classe <return>
.APPEND FROM Alunos <return>
```

CLASSE

NOME	SALA	SÉRIE
Alex	12	2
Marcio	15	3
Eduardo	12	2
Adelson	15	3

\+

ALUNOS

NOME	IDADE
Milton	7
Carlos	5
Bruno	6

=

CLASSE

NOME	SALA	SÉRIE
Alex	12	2
Marcio	15	3
Eduardo	12	2
Adelson	15	3
Milton		
Carlos		
Bruno		

BROWSE

BROWSE é um comando de tela, que permite a edição e a inclusão de tela de registros ao arquivo de dados ativo. O vídeo pode apresentar ao mesmo tempo um máximo de 17 registros, e tantos campos quanto couberem na tela.

Sintaxe:
BROWSE [FIELDS <lista de campos>]

Observações:
A opção FIELDS especifica os campos e a ordem em que eles serão mostrados. Para se assegurar de iniciar pelo primeiro registro entre antes do comando BROWSE com o comando GOTO TOP.

Veja também: APPEND, EDIT, CHANGE.

> NOTA
> Após o último registro, o comando o repetirá indefinidamente até que seja dado um ^Q ou ^W.

CANCEL

CANCEL termina a execução de um arquivo de comandos e retorna o dBASE II ao modo interativo de teclado.

Sintaxe:
CANCEL

Observações:
Em um arquivo comando (.cmd ou .prg), o texto que aparecer nas linhas seguintes ao CANCEL é ignorado pelo dBASE II. Esta área pode ser usada para comentários de programas.
CANCEL fecha todos os arquivos de comandos que estejam abertos.

Veja também: RETURN

CHANGE

CHANGE é usado para possibilitar edição em tela de campos e registros especificados do arquivo ativo.

Sintaxe:
CHANGE [<escopo>] [FIELD <lista de campos>] [FOR/WHILE <condição>]

Observações:
Todos os registros são editados sequencialmente, a não ser que restringidos pelo escopo ou pela cláusula FOR/WHILE.

Todos os campos são apresentados pela edição a não ser que restringidos pela cláusula FIELDS.

Exemplos:
O arquivo de banco de dados Cadastro será usado para o exemplo abaixo:

Para editar o Nome:

```
.USE CADASTRO <return>

.CHANGE FIELD NOME <return>

RECORD: 00001
NOME: Marcelo Vianna
TROCAR ? nn
PARA:n

RECORD: 00001
NOME: Marcelo Viana
TROCAR?
```

A saída deste comando se dá com "Return" vazios ou "ESC" para retornar ao "prompt". A opção ALL troca todos os itens.

Veja também: EDIT, MODIFY COMMAND, SET FORMAT TO

CLEAR

CLEAR fecha os arquivos de dados abertos, libera todas as variáveis de memória e seleciona a área primária de trabalho.

Sintaxe:
CLEAR

Observações:
Fechando um banco de dados também serão fechados os arquivos de índice (.NDX), de formatação (.FMT) associados a ele.

Veja também: RELEASE ALL

CLEAR GETS

CLEAR GETS causa a subsequente declaração READ ignorar quaisquer @...GETs emitido antes da execução do CLEAR GETS.

Sintaxe:
CLEAR GETS

Observações:
O dBASE II permite um máximo de 64 @...GETS antes que um CLEAR GETS seja emitido.

CONTINUE

CONTINUE retorna a procura iniciada pelo comando LOCATE. CONTINUE procura pelo próximo registro no arquivo de dados ativo que satisfaça a condição especificada pelo último comando LOCATE emitido.

Sintaxe:
CONTINUE

Observações:
CONTINUE termina quando um registro que satisfaça a condição especificada é encontrado, ou quando o fim do escopo LOCATE original é alcançado, qualquer dos dois que ocorra primeiro.

Se a procura for bem sucedida, o ponteiro de registro é posicionado no registro que satisfaz a condição especificada.

Se a procura não for bem sucedida, o ponteiro de registro é posicionado no último registro do escopo do comando LOCATE original. Se este é o último registro do banco de dados, o flag de fim-de-arquivo é setado para verdadeiro.

A indicação visual de um CONTINUE bem sucedido é a mostra do número do registro. Uma tentativa fracassada é indicada pela mensagem "Fim de arquivo encontrado".

Exemplo:
O banco de dados Cadastro será usado para o exemplo a seguir:

Para ilustrar o uso do CONTINUE, os registros contendo MG ou SP no campo Estado vão ser localizados:

```
.USE Cadastro <return>
.LOCATE FOR ESTADO = "MG" .OR. ESTADO = "SP"
RECORD: 00003
.DISP <return>
00003 Evaldo Lemos R.do Ouvidor 275 B.Horizonte MG 30000
.CONTINUE <return>
RECORD: 00005
.DISP <return>
00005 Ricardo Noschese Av. Ibirapuera 945 São Paulo SP 01510
.CONTINUE <return>
```

```
RECORD:00009
.DISP <return>
00009 Neander Toledo Av. Paulista 10000 São Paulo SP 01310
.CONTINUE <return>
FIM DE ARQUIVO ENCONTRADO
```

Veja também: LOCATE

COPY

COPY copia todo ou parte do arquivo ativo para um arquivo receptor designado. Se o arquivo destino já existe, ele vai ser anulado pelos novos dados. Se não existe, ele vai ser criado pelo comando COPY. COPY é o comando básico para gerar dados para programas fora do dBASE II.

Sintaxe:
COPY TO <nome do arqruivo> [<escopo>] [FIELDS <lista de campos>] [FOR/WHILE <condição>] [SDF.DELIMITED [WITH <delimitador>]

Observações:
Todos os registros não marcados para deleção serão copiados para o arquivo-destino, a não ser que restringidos pelo escopo ou pela cláusula FOR/WHILE. Todos os campos serão copiados para o arquivo-destino, a não ser que restringidos por FIELDS <lista de campos>.
Se a opção SDF/DELIMITED é usada, a extensão do arquivo default será .txt. Se não, a extensão do arquivo default será .dbf.
Se a opção DELIMITED é usada, o delimitador default será aspas duplas para campos de caracter. Todos os campos são separados por vírgulas.

Exemplo:
O banco de dados Alunos contém o nome, sala e classe para diversos alunos numa escola. Para criar um novo arquivo que contenha somente os campos Nomes e Sala para todos secundaristas:

```
.USE Alunos <return>
.COPY TO ALUNOS_2 FIELDS Nome, Sala, FOR Classe = "2" <return>
```

ALUNOS

NOME	SALA	CLASSE
ALBÉRIO	12	2
JONAS	15	3
SÉRGIO	12	2
TIAGO	15	3
ANDERSON	15	3
RAFAEL	12	2

ALUNOS_2

NOME	SALA
ALBÉRIO	12
SÉRGIO	12
RAFAEL	12

Veja também: SET DELETED

DATALOGICA

COPY STRUCTURE

COPY STRUCTURE copia ssomente a estrutura do arquivo ativo para o arquivo destino designado. Se o arquivo destino já existir, será regravado. Se ele não, existir, será criado.

Sintaxe:
COPY STRUCTURE TO <nome do arquivo> [FIELDS <lista de campos>]

Observações:
O nome do arquivo deve incluir a designação do drive, se não for o drive default.
A não ser que outra seja especificada, a extensão do arquivo destino será .dbf.
A estrutura inteira será copiada a não ser que a opção FIELDS seja usada.

COUNT

COUNT conta o número de registros no arquivo de dados ativo que satisfazem uma condição especificada.

Sintaxe:
COUNT [<ESCOPO>] [FOR/WHILE <condição>] [TO <varmem>]

Observações:
A não ser que restringidos pelo escopo ou pela cláusula FOR/WHILE, todos os registros serão contados.

Exemplos:
O banco de dados Cadastro será usado nos exemplos abaixo.
Para contar o número de registros no banco de dados:

```
.USE cadastro <return>
.COUNT <return>
CONTAGEM=00010
```

Para determinar o número de registros quando o campo NOME começar com a letra R; e salvar o resultado na variável de memória Letra.

```
.COUNT FOR $(NOME,1,1)="R" TO Letra <return>
CONTAGEM=00002
.? Letra <return>
2
```

CREATE

CREATE define a estrutura de um novo arquivo de dados e o adiciona ao diretório do disco.

Sintaxe:
CREATE <nome do arquivo>

Observações:
A extensão do arquivo .dbf vai ser assumida a não ser que outra seja especificada.
Os itens abaixo devem ser definidos para cada campo do novo arquivo de dados.

Nome do campo
Tipo
Tamanho
Casas decimais

O número de casas decimais deve ser especificado apenas para campos numéricos. O ponto decimal (se usado) conta como um byte no tamanho do campo.
As instruções completas de mensagens de erro aparecem na parte inferior da tela.
O tipo pode ser especificado pelo uso da primeira letra dos tipos de dados (C,N,L).
O processo CREATE (cria) é finalizado pressionando-se <RETURN> quando pronto para definir um novo campo.

DELETE

DELETE marca os registros que devem ser eliminados do arquivo de dados ativo.

Sintaxe:
DELETE [<escopo>] [FOR/WHILE <condição>]

Observações:
A menos que seja especificados pelo escopo ou pela cláusula FOR/WHILE; somente o registro atual será marcado para eliminação. Nos bancos de dados, registros marcados para eliminação são indicados por um asterisco (*) na primeira posição do registro.

Exemplos:
O banco de dados Cadastro é usado para os seguintes exemplos.

Para marcar somente o primeiro registro no banco de dados para eliminação:

```
.USE CADASTRO
.DELETE <return>
00001 APAGADO(S)
```

Para marcar o registro 10 para deleção:

```
.DELETE RECORD 10
00001 APAGADO(S)
```

Veja também: PACK, RECALL, SET DELETED ON

DELETE FILE

DELETE FILE remove um arquivo do diretório do disco.

Sintaxe:
DELETE FILE <nome do arquivo>

Obsevações:
O nome do arquivo deve conter a extensão e a designação do drive se outra que não a do drive default.
Este comando não pode ser usado para eliminar um arquivo aberto.

Veja também: USE

DISPLAY

DISPLAY é usado para mostrar o conteúdo de um arquivo de dados.

Sintaxe:
DISPLAY [OFF] [<escopo>] [<list exp>] [FOR/WHILE <condição>] [TO PRINT]

Observações:
Será mostrado o registro atual, a não ser que outro seja especificado pelo escopo ou pela cláusula FOR/WHILE.
Todos os campos são mostrados, a não ser que restringidos pelas cláusulas.
É mostrado o número de registros, a não ser que desativado pela opção OFF.
Se mais de 20 registros são mostrados, DISPLAY faz uma pausa a cada 20 linhas, e mostra a mensagem "Aperte qualquer tecla para continuar..."

DISPLAY MEMORY

DISPLAY MEMORY mostra o nome, o tipo e o tamanho de cada variável de memória. Também mostra o número de variáveis de memória ativas, o número possível de variáveis adicionais, a quantidade de memória usada, e a quantidade de memória ainda disponível para variáveis de memória.

Sintaxe:
DISPLAY MEMORY [TO PRINT]

Observações:
O número máximo de variáveis de memória ativa é 64. O número possível de variáveis de memórias adicionais é 64.
A quantidade máxima de memória que pode ser usada para variáveis de memória é 1536 bytes.
Se o número de variáveis de memória ativas excede uma tela cheia, DISPLAY MEMORY vai entrar em pausa após cada tela com a mensagem "Pressione qualquer tecla para continuar...".
A mostra de variáveis numéricas contém as duas representações: o número como vai ser mostrado na tela, e o seu formato interno. Pode haver uma grande diferença entre os dois formatos.

Exemplos:
Para ilustrar, a string de caracter "Ola" foi guardada na variável Saudacoes; um True lógico (.T.) foi guardado na variável Certo e o número 400 foi guardado na variável Numero.

```
.STORE "OLA" TO SAUDACOES
OLA
.STORE T TO CERTO
.T.
.STORE 400 TO NUMERO
.DISPLAY MEMORY
SAUDACOES (C) OLA
CERTO (L) .T.
NUMERO (N) 400
** TOTAL ** 03 VARIÁVEIS 00013 BYTES
```

DISPLAY STATUS

DISPLAY STATUS, proporciona uma visão geral da situação atual do processamento.

Sintaxe:
DISPLAY STATUS

Observações:
As seguintes informações são fornecidas para cada arquivo de dados abertos:

- Nome do arquivo
- Número de área de trabalho
- Nomes dos arquivos de índices abertos
- Relação de bancos de dados
- Chave índice para cada arquivo índice aberto

Outras informações são fornecidas por esse comando:

- Caminho de pesquisa para o arquivo corrente
- Drive default
- Situação atual de todos os comandos SET ON/OFF

DISPLAY STRUCTURE

DISPLAY STRUCTURE dá as seguintes informações sobre o arquivo de dados ativo: nome do arquivo, número de registros, a última data em que qualquer item foi alterado, a definição completa de cada campo e o número total de bytes por registro.

Sintaxe:

DISPLAY STRUCTURE (TO PRINT)

Observações:

O número de campos excede 16, DISPLAY STRUCTURE vai entrar em pausa depois de cada 16 nomes de campos com a mensagem "Pressione qualquer tecla para continuar..."

Exemplo:

O exemplo seguinte usa o banco de dados Cadastro.

```
.USE Cadastro <return>
.DISPLAY STRUCTURE <return>
Structure for file      : A:Cadastro.dbf
Number of Records:      00010
Date of last update:  07/16/85
Primary Use Database
CPO NOME
001  NOME                           TIPO   TAM        DEC
002  ENDERECO                         C    020
003  CIDADE                           C    025
004  ESTADO                           C    012
005  CEP                              C    002
**                                    C    005
                        TOTAL         **   065
```

NOTA

A extensão é sempre o total de bytes mais 1 de controle

DO

DO faz com que um arquivo de comandos (.prg ou .cmd) do dBASE II seja executado, permitindo a passagem de parâmetros necessários ao programa chamado.

Sintaxe:
DO (nome do arquivo)

Observações:
O nome do arquivo deve incluir a designação do drive se não for o drive default. Uma extensão .prg ou .cmd é assumida a não ser que outra seja especificada.

Cada DO (nome do arquivo) conta como um arquivo aberto. O número total de arquivos abertos (arquivos de dados, de índices, de formatação, de comando, e assim por diante) não pode ser mais que 15.

Quando termina o programa chamado pelo DO, o controle retorna para o programa que chamou, ou para o teclado se o DO foi emitido do teclado.

DO CASE

DO CASE é um comando de programação estruturada que seleciona apenas uma ação entre várias alternativas.

Sintaxe:
```
DO CASE
    CASE <Condição>
        <comandos>
    [CASE <Condição 1>]
    [OTHERWISE]
        <comandos>
ENDCASE
```

Observações:
DO CASE deve ser finalizado por um ENDCASE.

Pares de comandos como DO CASE...ENDCASE, IF...ELSE...ENDIF, e DO WHILE...ENDDO podem estar contidos dentro do DO CASE. São permitidos CASEs encadeados.

O primeiro CASE <condição> que for satisfeito será selecionado.

Após um CASE verdadeiro ser encontrado e seus comandos associados serem processados, nenhum outro CASE será avaliado. Nunca serão selecionados dois CASEs.

O OTHERWISE opcional é interpretado como significando todos os outros casos não especificados.

Veja também: DO, DO WHILE, IF, MODIFY COMMAND

DO WHILE

DO WHILE é um comando de programação estruturada que permite que os comandos entre o DO WHILE e o ENDDO sejam repetidos enquanto uma condição especificada for verdadeira.

Sintaxe:

```
DO WHILE <condição>
   <comandos>
ENDDO
```

Observações:

Quaisquer comandos estruturados dentro de uma estrutura DO WHILE...ENDDO devem estar com seus respectivos pares. São permitidos DO WHILEs encadeados. Comentários digitados na mesma linha do ENDDO são ignorados. Se a condição DO WHILE for verdadeira, os comandos entre o DO WHILE e o ENDDO serão executados sequencialmente. Quando o ENDDO for alcançado, o processo retorna ao DO WHILE. Se a condição ainda for verdadeira, o processo será repetido. Quando se torna falsa, o programa processará os comandos abaixo do ENDDO.

Veja também: LOOP, EXIT

EDIT

EDIT é um comando de tela usado para alterar o conteúdo de um registro no arquivo de dados ativos.

Sintaxe:
EDIT [<expN>]

Exemplo:
O arquivo de dados Cadastro foi usado no exemplo abaixo:

Para editar o quarto registro no banco de dados:

```
.EDIT 4 <return>
RECORD # 00004
NOME        :Roberto Amorim :
ENDERECO :  Quadra K 931 :
CIDADE   :  Brasília :
ESTADO   :  DF:
CEP      :  70000:
```

Veja também: MODIFY COMMAND, SET FORMAT TO

ERASE ou ? CHR(12)

ERASE limpa a tela e posiciona o cursor no canto superior esquerdo (coordenadas 0,0).

> **NOTA**
> Estes comandos só funcionam para 40 colunas (sem o cartão de 80 colunas).
> Para o uso com o cartão de 80 colunas o comando ERASE deverá ser substituído por ? CHR(2)

FIND

FIND procura o primeiro registro em um arquivo de dados **indexado**, cuja chave seja igual à string especificada. O comando FIND é um meio rápido de pesquisa.

Sintaxe:
FIND <string de caracter>

Observações:
A string de caracter não necessita ser delimitado, mas deve começar pelo início da chave.
Se chave começa com brancos, a string especificada (incluindo o número de brancos) deve ser delimitado por aspas simples ou duplas.
FIND funcionará somente quando o primeiro caracter ou caracteres numa chave forem especificados. Quando o conteúdo a ser pesquisado estiver em uma variável de memória, a variável deve ser usada com a função & (MACRO).
Se a string não for encontrada, aparecerá na tela a mensagem NO FIND, Neste caso, o flag fim de arquivo é setado para True (verdadeiro).

Exemplos:
O arquivo Cadastro, indexado pelo campo Nome, é usado para os exemplos abaixo.

Para achar o primeiro registro com o NOME começando por C:

```
.FIND C <return>
.? NOME <return>
Cesar Sampaio
```

Para achar o registro para Zychowoski (não existe este registro):

```
.FIND Zychowoski <return>
Não encontrado
```

Veja também: INDEX, SET INDEX, USE

GO/GOTO

GO/GOTO posiciona o ponteiro de registro em um determinado registro no arquivo de dados ativo.

Sintaxe:
GO ou GOTO (expN) /BOTTOM/TOP

Observações:
Se um arquivo índice estiver em uso com o banco de dados, TOP e BOTTOM referem-se respectivamente ao primeiro e último registro no arquivo índice.

Exemplos:
O banco de dados Cadastro foi usado para os exemplos abaixo para posicionar o ponteiro no registro 8:

```
.USE Cadastro <return>
.GOTO 8 <return>
.? # <return>
8
```

Para posicionar o ponteiro no registro 10:

```
.STORE 8 TO Nnum <return>
8
.GO (Mnum + 2) <return>
.? # <return>
10
```

Para posicionar o ponteiro de registro no começo do arquivo de dados:

```
.GOTO TOP
.DISP
00001 Marcelo Vianna R.  15 de Novembro 1056 Manaus AM 69000
```

HELP

HELP é um tutorial em tela, orientado por menu, que fornece informações sobre os comandos do dBASE II e seu uso.

Sintaxe:
HELP
HELP (palavra-chave)

Observações:
O comando HELP usa o arquivo DBASEMSG.TXT do disco fornecido pela DATALOGICA. A palavra-chave deve ser um comando ou função do dBASE II.

Exemplos:
Para ativar o recurso HELP do dBASE II

```
.HELP <return>
```

Isto mostra o menu do HELP.

Para ativar o HELP para o comando COPY:

```
.HELP COPY <return>
```

IF

IF é um comando de programação estruturada que possibilita o processamento condicional de comandos. Ele deve terminar com ENDIF.

Sintaxe:

```
IF <expressão>
        <comandos>
[ELSE]
        <comandos>
ENDIF
```

Observações:
Qualquer comando estruturado dentro de uma estrutura IF...ENDIF deve estar adequadamente emparelhado. São permitidos IFs encadeados. O espaço seguinte ao ENDIF em uma linha pode ser usado para comentários.

Veja também: DO CASE

INDEX

INDEX cria um arquivo no qual os campos chave de um arquivo de dados associados são classificados em ordem alfabética, cronológica ou numérica.

O arquivo índice contém o campo chave e o número do registro correspondente de cada registro do arquivo.

Sintaxe:
INDEX ON <expressão chave> TO <nome do arquivo>

Observações:
A não ser que especificados de outro modo no nome do arquivo, serão assumindos o drive default e uma extensão .NDX.

A chave de índice pode ser um campo, ou uma expressão que envolva um até sete campos.

CAMPOS LÓGICOS NÃO podem ser usados na expressão da chave de índice.

A CLASSIFICAÇÃO por default é em ordem crescente.

OS REGISTROS DO ARQUIVO DE DADOS NÃO são reorganizados fisicamente pela operação INDEX.

Quando a chave índice consiste de mais de um campo o ITEM MAIS IMPORTANTE DEVE SER ESPECIFICADO PRIMEIRO.

CAMPOS NUMÉRICOS podem ser associados com campos do tipo caracter em uma operação e indexação, se forem usados com a função STR.

O COMPRIMENTO MÁXIMO de uma chave de índice é de 100 caracteres

Exemplos:
Indexar os nomes no arquivo de dados Cadastro.
O campo que contém os nomes foi chamado de Nome. O nome do arquivo Índice será Cadastro.

```
.USE Cadastro
.INDEX on Nome TO Cadastro
00010 RECORDS INDEXED
.DISPLAY NOME
00007 Cesar Sampaio
00008 Eduardo Ponte
00003 Evaldo Lemos
00001 Marcelo Vianna
00010 Marcio Cunha
00009 Neander Toledo
00005 Ricardo Noschese
00004 Roberto Amorim
00002 Victor Lima
00006 Wagner Dutra
```

Veja também: FIND, REINDEX, SET INDEX, USE

INPUT

INPUT é usado basicamente em arquivos de comandos (programas) para permitir ao usuário uma entrada via teclado, terminada por <RETURN>.

Sintaxe:
INPUT [<mensagem>] TO <varmem>

Observações:
A mensagem pode ser tanto uma variável do tipo caracter como uma string literal. Se uma string de caracteres for usado, a mensagem deve ser limitada por "aspas simples", "aspas duplas", ou [colchetes].

Qualquer expressão válida do dBASE II pode ser digitada em resposta ao comando INPUT. O tipo de expressão digitada determina o tipo de variável criada.

Se uma expressão inválida ou um <RETURN> for entrado em resposta ao comando INPUT, um erro de sintaxe será indicado.

Exemplos:
Para indicar ao usuário "Digite valor da Renda:" e armazenar a entrada do teclado em uma variável de memória Renda.

```
.INPUT "Digite valor da Renda:" TO Renda
```

Na tela irá aparecer:

```
Digite Valor da Renda
```

Veja também: ACCEPT, STORE, WAIT

INSERT

INSERT adicona um único registro novo, ao arquivo de dados, na posição do registro atual.

Sintaxe:
INSERT [BLANK] [BEFORE]

Observações:
A cláusula BEFORE insere um novo registro logo antes do registro atual. Se o registro atual for o registro número 5, INSERT BEFORE irá criar um novo registro 5; o velho registro 5 passará a ser o registro 6, e assim por diante.

Se a cláusula BEFORE for omitida, o novo registro será inserido imediatamente após o registro atual. Por exemplo, se o registro corrrente é o registro 5, INSERT criará um novo registro 6; o velho registro 6 passará a ser registro 7, a assim por diante.

Se a opção BLANK for omitida, INSERT apresentará o novo registro para digitação dos dados em tela. Os dados poderão ser digitados somente para este registro.

Exemplos:
O arquivo de banco de dados Cadastro foi usado no exemplo abaixo. Para inserir um novo registro imediatamente antes do registro 9 (isto é, criar um novo registro 9):

```
.USE Cadastro
.GOTO 9
.INSERT BEFORE
```

Imediatamente aparecerá no seu vídeo:

```
REGISTRO #          00009
NOME                     :                    :
ENDERECO            :                              :
CIDADE              :              :
ESTADO              :         :
CEP                 :              :
```

Veja também: @, APPEND, CHANGE, EDIT, MODIFY COMMAND, SET FORMAT

JOIN

JOIN cria um novo arquivo de dados, unindo registros e campos especificados de dois arquivos de dados existentes,

Sintaxe:
JOIN TO **<nome do arquivo>**
FOR **<condição>** [FIELDS **<lista de campos>**]

Observações:
O nome do arquivo destino deve incluir a designação do drive se não for o drive default. A extensão do arquivo .dbf é assumida, a não ser que outra seja especificada.

o arquivo ativo é JOINed (juntando) com um arquivo aberto da outra área de trabalho. A lista de campos pode conter campos de ambos os arquivos fonte. Campos do segundo arquivo são indicados pela sintaxe.

Se nenhuma lista de campo é especificada, assinala-se do primeiro arquivo ativo. Depois, os campos são transferidos do segundo arquivo até que o limite de 32 campos seja alcançado.

O ponteiro de registro é setado no primeiro registro do arquivo ativo. Cada registro do segundo arquivo é avaliado pelo FOR **<condição>**. Se a condição especificada for verdadeira, um novo registro é adicionado ao arquivo de destino. Quando todos os registros no segundo arquivo forem examinados, o arquivo ativo avança um registro e o processo se repete. Isto continuará até todos os registros no arquivo ativo serem processados. Este processo sequencial pode consumir muito tempo, quando se lida com arquivos grandes.

Deve-se tomar cuidado quando utilizar este processo. É possível que dois arquivos de banco de dados quando agrupados façam com que o arquivo destino exceda o espaço disponível no disco.

Exemplo:
A ilustração abaixo descreve como juntar o arquivo Alunos ao arquivo Profess para formar o novo arquivo Classe:
Note que S. indica o arquivo Secundário.

```
.USE ALUNOS
.SELECT SECONDARY
.USE PROFESS
.SELECT PRIMARY
.JOIN TO Classe FOR Sala=S.Sala Fields Nome, Profess
```

ALUNOS

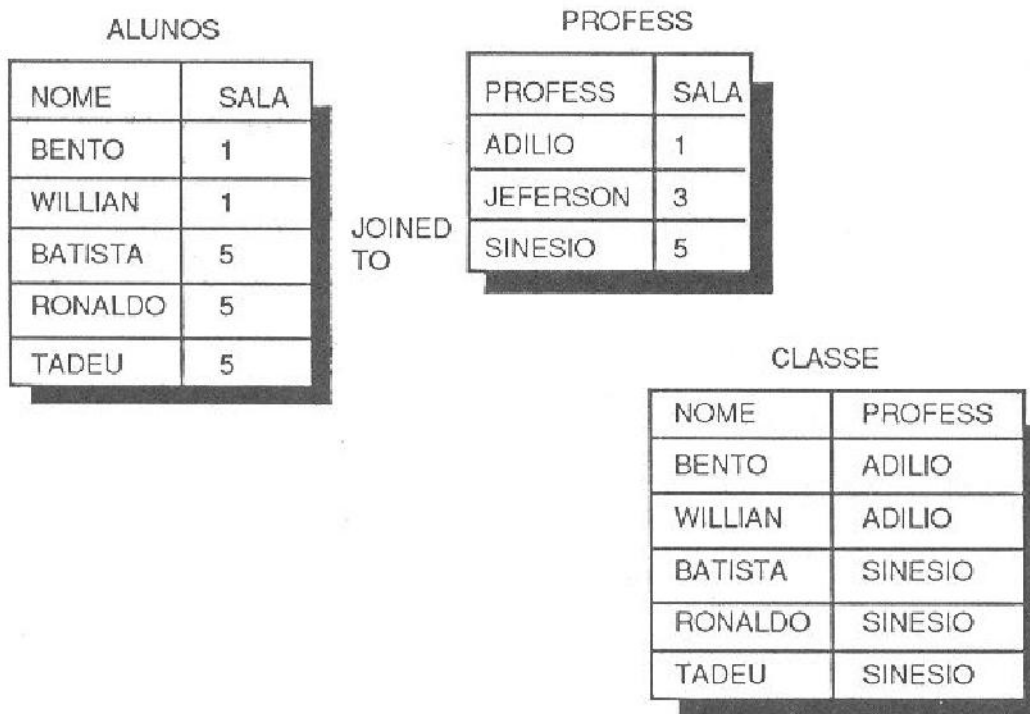
NOME	SALA
BENTO	1
WILLIAN	1
BATISTA	5
RONALDO	5
TADEU	5

JOINED TO

PROFESS

PROFESS	SALA
ADILIO	1
JEFERSON	3
SINESIO	5

CLASSE

NOME	PROFESS
BENTO	ADILIO
WILLIAN	ADILIO
BATISTA	SINESIO
RONALDO	SINESIO
TADEU	SINESIO

Veja também: SET LINKAGE

LIST

LIST é usado para ver o conteúdo de um arquivo de dados.

Sintaxe:
LIST [OFF] [<escopo>] [**<lista de expressões>**]
[FOR/WHILE **<condição>**]
[TO PRINT]

Observações:
LIST é idêntico ao comando DISPLAY, exceto que:
• Não há uma pausa periódica na listagem
• Todos os registros são mostrados, a não ser que alternativas sejam especificadas.

Veja também: DISPLAY, SET HEADING, SET MARGIN

LIST FILE

LIST FILE mostra o conteúdo no drive indicado. Usando os parâmetros opcionais, esse comando pode também ser usado para mostrar os nomes de quaisquer ou de todos os arquivos no drive indicado.

Sintaxe:
LIST FILE [ON] [<drive>] [<LIKE <máscara>]

Observações:
A não ser que seja especificado de outro modo, LIST FILE mostra somente a informação sobre os arquivos de dados do drive default.

Exemplos:
Para mostrar informações sobre o arquivo de dados:

```
.LIST FILE
```

Para mostrar todos os nomes dos arquivos dbf do disco:

```
.LIST FILE LIKE *.*
```

Para mostrar os nomes dos arquivos-índice:

```
.LIST FILE LIKE *.NDX
```

Para mostrar os nomes dos arquivos-índice que comecem com D:

```
.LIST FILE LIKE D*.NDX
```

Para mostrar os nomes dos arquivos que têm 5 caracteres de tamanho, onde D é o terceiro caracter:

```
.LIST FILE LIKE ??D??.*
```

Para mostrar todos os nomes dos arquivos iniciados por Alunos:

```
.LIST FILE LIKE Alunos *.*
```

LIST MEMORY

LIST MEMORY mostra o nome, tipo e tamanho de cada variável de memória. Mostra também o número de variáveis de memória ativas, quantidade de memória usada, e a quantidade de memória ainda disponível para variáveis de memória.

Sintaxe:
LIST MEMORY [TO PRINT]

Observações:
LIST MEMORY é idêntico ao comando DISPLAY MEMORY, exceto que não há pausa periódica na listagem.

LIST STATUS

LIST STATUS fornece uma visão geral da situação atual do processamento.

Sintaxe:
LIST STATUS [TO PRINT]

Observações:
LIST STATUS é idêntico ao comando DISPLAY STATUS, exceto que não há uma pausa periódica na listagem.

LIST STRUCTURE

LIST STRUCTURE mostra as seguintes informações sobre o arquivo de dados ativo: o nome do arquivo, o número de registros, a última data em que qualquer dados foi atualizado, a definição completa de cada campo, e o número total de bytes por registro.

Sintaxe:
LIST STRUCTURE [TO PRINT]

Observações:
LIST STRUCTURE é idêntico ao comando DISPLAY STRUCTURE, exceto que não há uma pausa periódica na listagem.

LOCATE

LOCATE procura sequencialmente no arquivo ativo um registro que satisfaça a condição especificada.

Sintaxe:
LOCATE [<escopo>] FOR <condição>

Observações:
A não ser que restringido pelo escopo, o comando LOCATE pesquisará o banco de dados inteiro, começando pelo primeiro registro. O uso da opção NEXT n para o escopo limita a procura a n registros, começando pelo registro atual.

Se a procura não tiver sucesso e o fim-de-arquivo não for alcançado, o flag de fim-de-arquivo não será ativado. Não existe flag para: "Fim-do-escopo-do-LOCATE".

A indicação visual de um LOCATE bem sucedido é a mostra do número do registro. Uma tentativa sem sucesso é indicada pela mensagem "Fim do arquivo encontrado".

Exemplos:
O banco de dados Cadastro foi usado para o exemplo abaixo:

Para localizar o primeiro registro contendo Porto Alegre no campo cidade:

```
.USE Cadastro
.LOCATE FOR cidade="Porto Alegre"
RECORD: 00010
.DISP
00010 Marcio Cunha R. da Consolacao 963 Porto Alegre RS 90000
.CONTINUE
FIM DE ARQUIVO ENCONTRADO
```

Veja também: CONTINUE, EOF

LOOP

LOOP é usado para retornar ao início de uma estrutura DO WHILE...ENDDO. É equivalente ao ENDDO. O comando é normalmente usado para evitar que haja a execução dos comandos restantes na construção DO WHILE, quando uma condição especial é encontrada.

Sintaxe:
LOOP

Observações:
Para funcionar como itencionado, o comando LOOP deve estar dentro de uma estrutura IF...ENDIF ou DO CASE...ENDCASE.

Veja também: DO, DO CASE...ENDCASE, DO WHILE...ENDDO, EXIT, IF...ENDIF.

MODIFY COMMAND

MODIFY COMMAND ativa o editor de texto do dBASE II. Sua função principal é criar e editar arquivos de comando e de formatação, mas também pode ser usado com qualquer arquivo de texto padrão ASCII.

Sintaxe:
MODIFY COMMAND <nome do arquivo>

Observações:
Se nenhuma designação do drive ou extensão for especificada como parte do nome do arquivo, o drive default e a extensão .prg serão assumidas.

Quando este comando é emitido, o dBASE II procura o arquivo.

Se existir, o arquivo será chamado para edição. Cada vez que um arquivo é editado, a versão prévia é salva como um arquivo de reserva (backup), com extensão .bak. Se o arquivo não for achado, um novo arquivo será criado.

Para imprimir um arquivo criado pelo MODIFY COMMAND, saída do dBASE II você terá de fazê-lo através do type do Sistema Operacional.

Exemplo:
A>TYPE ARQUIVO.PRG <CTRL-P>
A>TYPE ARQUIVO.CMD <CTRL-P>

O tamanho máximo de arquivo que o MODIFY COMMAND pode manipular não tem limite, lembrando que um programa muito extenso poderá acarretar alguns problemas, como por exemplo a perda de parte de seu arquivo .cmd ou .prg.

MODIFY COMMAND <NOME DO ARQUIVO>.FRM

Neste caso o MODIFY COMMAND é utilizado para a modificação de um arquivo relatório (.FRM).

Exemplo:

```
.MODIFY REPORT CADASTRO.FRM

M=0, L=50, W=80
Y
DADOS PESSOAIS; =================
N
N
20,NOME
NOME;_______________
25,ENDERECO
ENDERECO;_______________
12,CIDADE
CIDADE;__________
12,ESTADO
ESTADO;__________
6,CEP
CEP;__________
```

Após encerrada as modificações você teclará Control-W para gravá-las ou Control-Q para abortá-las.

MODIFY STRUCTURE

MODIFY STRUCTURE é usado para modificar a estrutura de um arquivo de dados existente.

Sintaxe:
MODIFY STRUCTURE <nome ao arquivo>

Observações:
Um arquivo backup (cópia de reserva) tem que ser feito antes da execução.

O seguinte deve ser especificado para cada campo para que possa pertencer ao banco de dados:

Nome do Campo
Tipo
Tamanho
Casas Decimais

O NÚMERO DE CASAS DECIMAIS deve ser fornecido somente para os campos numéricos. Se usado, o ponto decimal conta como byte no tamanho do campo.

O CAMPO ATUAL é indicado pelo vídeo reverso, ou pela presença do cursor.

INSTRUÇÕES completas e MENSAGENS DE ERRO aparecem na parte inferior da tela.

O TIPO DE CAMPO pode ser fornecido usando a primeira letra do tipo de dado (C, N, L).

O CURSOR não pode avançar além de um campo ainda incompleto.

UM CAMPO PODE SER INSERIDO posicionando-se o cursor onde a inserção deve ser feita e pressionando a tecla CTRL-N.

UM CAMPO PODE SER DELETADO posicionando-se o cursor no campo a ser deletado e pressionando a tecla CTRL-T.

O PROCESSO DE MUDANÇA É TERMINADO pressionando-se <RETURN> ao definir um novo campo ou através da tecla CTRL-W

Observações importantes:
ANTES DE MODIFICAR A ESTRUTURA DE SEU ARQUIVO SALVE OS DADOS COPIANDO-OS PARA UM OUTRO ARQUIVO.LOGO DEPOIS QUE VOCÊ MODIFICAR SUA ESTRUTURA, VOCÊ IRÁ RETOMÁ-LOS NOVAMENTE ATRAVÉS DO COMANDO APPEND FROM, LEMBRANDO QUE SE VOCÊ FOR MODIFICAR O NOME DEALGUM CAMPO TERÁ DE SER USADA A CLÁUSULA .SDF, TANTO COM O COPY QUANTO COM O APPEND FROM.

Exemplo:

```
.USE CADASTRO
.COPY TO SALVA .SDF
.MODIFY STRUCTURE
.APPEND FROM SALVA .SDF
```

*

Um asterisco (*) no início de uma linha de comando num arquivo de comando indica que a linha não deve ser executada.
Usa-se este recurso para inserir comentários dentro do arquivo.

Sintaxe:
*** <texto>**

Observações:
Se uma linha de * termina com ponto e vírgula, o dBASE II lê a próxima linha como parte da linha de comentário.

Exemplo:

```
*Este loop somente toma tempo
STORE 1 TO X
DO WHILE X < 100
STORE X + 1 TO X
ENDDO
```

Veja também: MODIFY COMMAND

PACK

PACK elimina os registros que estão marcados para deleção em um arquivo de dados ativo.

Sintaxe:
PACK

Observações:
Todos os arquivos de índice abertos serão automaticamente REINDEXADOS. O espaço do disco usado pelos registros deletados será liberado para o sistema operacional quando o arquivo de banco de dados for fechado.
As informações de diretório que dizem respeito ao arquivo não serão corrigidas até que o arquivo seja fechado.

Veja também: DELETE, RECALL, REINDEX.

QUIT

QUIT fecha todos os arquivos abertos, termina a sessão do dBASE II, e volta para o sistema operacional.

Sintaxe:
QUIT

Observações:
Este é o único método seguro para sair do dBASE II.

QUIT TO (CP/M)

QUIT
QUIT [TO **<com file list>**]

Este comando fecha todos os arquivos de banco de dados, arquivos de comandos, e arquivos alternativos e retorna o controle para o sistema de operação.

A mensagem ***FIM DO dBASE*** é mostrada.

Se a palavra TO é incluída, então todos os programas na **<com file list>** serão executados em sequência pelo CP/M. Esta característica deixa você sair do dBASE e encadear-se com outros "softwares".

Não há limite para o número de programas ou comandos CP/M que podem ser executados desde que o limite de 254 caracteres por comando não seja excedido. O dBASE pode ser reproduzido no final da string de comandos.

Entretanto, não é exigido; o CP/M receberá o controle quando a string de comandos for finalizada.

Exemplo:

```
.QUIT TO "DIR B:";"PIP PRN: = ALTERNAT.TXT", "DBASE"
```

Neste exemplo, o dBASE é retirado, um catálogo do drive-b é feito PIP é então solicitado a copiar um arquivo para o equipamento de impressão, e o dBASE é reintroduzido.

> **NOTA**
> Para MP/M as versões sob o sistema operacional DOS (MSX-DOS, HB-DOS e MS-DOS) não é possível a execução do comando "QUIT TO".

READ

READ ativa todos os @...GETs emitidos desde o último CLEAR ou CLEAR GETS. O comando READ é comumente usado em arquivos de comando para entrada ou edição de dados em tela.

Sintaxe:
READ

Observações:
READ usa as teclas padrão de controle do cursor.

Exemplos:
Use o comando READ para editar o conteúdo de uma variável de memória.

```
.STORE " " TOMnome
.@ 10,10 SAY "Digite seu nome:
     "GET Mnome
.READ
```

READ coloca o cursor dentro do Mnome, pronto para edição.

Veja também: CLEAR GETS, SET FORMAT TO

RECALL

RECALL recupera os registros do arquivo ativo que estão marcados para eliminação.

Sintaxe:
RECALL [<escopo>] [FOR/WHILE <condição>]

Observações:
A menos que seja especificado pelo escopo ou pela cláusula FOR/WHILE, somente o registro atual será recuperado.

RECALL não recupera registos que tenham sido removidos do banco de dados pelos comandos PACK.

Exemplos:
O banco de dados Cadastro é usado para os seguintes exemplos.

Para recuperar somente o primeiro registro do arquivo de dados (admitindo-se que os registros 1, 10 e 5 tenham sido marcados para deleção):

```
.USE Cadastro
.RECALL
00001 RECUPERADO(S)
```

Para reintegrar o registro 10:

```
.RECALL RECORD 10
00001 RECUPERADO(S)
```

Para reintegrar o registro 5 e todos os outros registros que tenham sido marcados para deleção:

```
.RECALL ALL
. n RECUPERADO(S)
```

Veja também: DELETE, PACK, SET DELETED ON.

REINDEX

REINDEX reconstrói todos os arquivos de índice ativos.

Sintaxe:
REINDEX

Exemplos:
O arquivo de dados Cadastro foi usado para ilustrar o comando:

```
.USE Cadastro
.SET INDEX TO Cadpess
.REINDEX
REINDEXANDO ARQUIVO INDICE CADPESS
00010 REGISTROS INDEXADOS
```

Veja também: INDEX, PACK, SET INDEX TO, USE.

RELEASE

RELEASE elimina variáveis de memória e abre espaço na memória para novos usos.

Sintaxe:
RELEASE [**<lista de varmem>**] [ALL[LIKE/EXCEPT **<máscara>**]]

Observações:
Na máscara, o ponto de interrogação (?) representa um caracter, e o asterisco (*) representa um ou mais caracteres.

Exemplos:
Para liberar (RELASE) todas as variáveis de memória que comecem com a letra m:

```
.RELEASE ALL LIKE M*
```

Para liberar todas as variáveis de memória exceto as que tiverem a letra x como terceiro caracter:

```
.RELEASE ALL EXCEPT ??X*
```

Para liberar todas as variáveis de memória:

```
.RELEASE ALL
```

Veja também: RESTORE, RETURN, SAVE, STORE

REMARK

REMARK [alguns caracteres]

Este comando proporciona a exposição de quaisquer caracteres. O conteúdo deste comando é mostrado no equipamento de saída quando este comando é encontrado.

Exemplos:

```
.REMARK ******REMARK TEST********
******REMARK TEXT********
```

RENAME

RENAME muda o nome de um arquivo.

Sintaxe:
RENAME <nome antigo do arquivo> TO <nome novo>

Observações:
Tanto o velho como o novo nome do arquivo devem incluir as extensões e os designadores de drive, se não for o drive default.

O novo nome do arquivo não pode ser o mesmo que o de um arquivo já existente no mesmo disco.

Um arquivo aberto não pode ser renomeado.

Exemplos:
Para renomear o arquivo de banco de dados Escola.dbf para CLASSES.dbf:

```
.RENAME Escola.dbf TO Classes.dbf
```

Veja também: USE

REPLACE

REPLACE é usado para mudar o conteúdo de campos especificados do arquivo de dados ativo.

Sintaxe:
REPLACE[<**escopo**>] <**campo**> WHITH
<**EXP**>
[,<**campo**> WHITH <**exp**>...]
[FOR/WHILE <**condição**>]

Observações:
A menos que seja especificado pelo escopo ou pela cláusula FOR/WHILE, somente o registro atual será alterado.

Se a alteração for feita em um campo indexado, o arquivo índice será atualizado.

Substituições múltiplas em campo indexado devem ser feitas com cuidado. Quando a substituição é realizada, o registro muda de lugar no arquivo índice. Por esta razão NÃO SE DEVE FAZER SUBSTITUIÇÕES MÚLTIPLAS EM CAMPO INDEXADO.

Exemplos:
Será usado o arquivo de dados Cadastro. A CIDADE quando for igual a São Paulo será substituída por cidade igual a Recife, observando que a função ! (uppercase) foi utilizado para que todas as cidades iguais a São Paulo sejam trocadas independente de estarem escritas em letras maiúsculas ou minúsculas.

```
.USE Cadastro
.REPLACE CIDADE WITH "RECIFE" FOR;
     (CIDADE) = "SÃO PAULO"
00002 SUBSTITUIÇÕES
```

Para aumentar o aluguel mensal em 10% e ajustar a comissão mensal:

```
.REPLACE ALL ALUGUEL WITH ALUGUEL
      * 1.1
n SUBSTITUIÇÕES
```

REPORT FORM

O REPORT é usado para preparar relatórios (tanto na tela como no papel) mostrando dados do arquivo em USO de uma maneira definida. Os Relatórios podem ter colunas entituladas, campos numéricos totalizados, e expressões mostradas envolvendo campos de dados, variáveis de memória, e constantes.

A frase FOR permite somente que as informações que correspondem ás condições da **<exp>** sejam reportadas; a frase TO PRINT envia o relatório à impressora tão bem quanto para a tela; e o **<escopo>** do relatório padroniza em ALL a menos que outro seja especificado.

A primeira vez que o comando REPORT é usado (para um novo relatório) um arquivo, FORM é construído. O dBASE prepara o usuário para especificação do formato do relatório e automaticamente gera o arquivo FORM. Subsequentes relatórios podem usar o arquivo FORM para evitar especificação do formato do mesmo.

Se a frase FORM do comando é omitida o usuário será preparado para o nome do arquivo form.

Se dá um erro no arquivo, REPORT FORM, o REPORT para de rodar e o dBASE II mostra a mensagem ERRO DE SINTAXE NA ESPECIFICAÇÃO DO FORMATO. Se o REPORT foi assumido no indicador de ponto, o controle retorna neste ponto. Se foi assumido de um arquivo de comando, a execução se resume com a linha de código seguinte o comando REPORT.

O exemplo a seguir indica as diversas operações do "Report Form". Pode ser controlado o número de espaços em branco para a margem esquerda (M – default 8), o número de linhas por página (L – default 57) e a centralização do cabeçalho na página (W – default 80).

A limitação para qualquer expressão entrada no REPORT FORM é 254 bytes. Isto inclui o cabeçalho do relatório, cabeçalhos das colunas, e conteúdo das colunas.

Se o usuário tentar entrar mais que 254 bytes, o gerador de Relatório simplesmente tenta ir ao item seguinte. No caso de cabeçalhos, é normalmente sucedida; mas no caso de conteúdo de colunas, não pode.

Especificamente, se a expressão que o usuário está entrando é sintaticamente correta até o ponto do 255 byte, ele vai ao item seguinte. Senão, ele dá a mensagem ao usuário:

```
ERRO DE SINTAXE, RE-ENTRE.
```

O REPORT pede pelo tamanho do campo a ser impresso e os conteúdos do campo. O tamanho pedido não tem relação com o tamanho real do campo a ser imprimido. Por exemplo um campo de arquivo que tem 23 caracteres de largura no relatório tem na realidade 20 caracteres de tamanho no banco de dados.

Os conteúdos das colunas podem ser campos de banco de dados, uma variável de memória, literais, ou expressões.

O significado dos caracteres especiais ";", "<" e ">" são:

; coloca o conteúdo a seguir na próxima linha embaixo do item anterior.

< ajusta o título à esquerda do campo.

> ajusta o título à direita do campo.

Outras opções no REPORT incluem totalização, subtotalização, e Relatórios resumidos. Nos relatórios resumidos, detalhe de registros não são mostrados, apenas totais e subtotais. Totalização e subtotalização são somente feitas em campos de natureza numérica. Só será solicitadas subtotalização se houver totalização.

Finalmente, um retorno de carro terminará o report form e começará a mostrar o relatório. Uma cópia será impressa na impressora se a frase TO PRINT for incluída no comando inicial.

Outros comandos do dBASE que afetam a operação do relatório são os "SET EJECT OFF", "SET HEADING TO" e o "SET DATE TO".

Antes do REPORT imprimir sua informação, ele ejeta uma página.

Esta capacidade pode ser suprimida com o comando SET EJECT OFF. O comando SET HEADING TO permite um cabeçalho adicional a ser adicionado no relatório quando da sua execução. Este comando tem um efeito para a duração de uma sessão. (O cabeçalho deve ser colocado cada vez que um novo dBASE é inicializado). O mesmo é para o comando SET DATE TO. A data do relatório pode ser mudada ou omitida pelo erro deste comando. Veja o comando SET para mais informações.

Chegará o tempo, quando esta capacidade não será mais adequada, formas, especiais deverão ser usadas, mais flexibilidade será desejada, com o formato de relatório, recuperação de dados dos bancos de dados requererá métodos mais complexos que o REPORT poderá suportar, etc.

Os comandos "@" e o SET FORMAT TO PRINT dará ao usuário mais poder sobre o formato do relatório. Veja o comando "@" para mais informações e exemplos.

No exemplo os hífens indicam grifo para os títulos nas linhas subsequentes. O mesmo apresenta um erro. O campo ESTADO terá seu hífem deslocado de uma linha. Isto se dá porque os dados são maiores que os itens. Basta aumentar a extensão para CIDADE para 3 que os hífens sairão na mesma linha. Os números à esquerda dos nomes dos campos servem então somente para centralização dos mesmos no relatório.

EXEMPLO DE COMO GERAR UM RELATÓRIO

```
.USE CADASTRO INDEX CADPESS
. REPORT FORM CADASTRO

ENTRE OPÇÕES, M = Margem esq, L = Linhas/pag., W = Tamanho pag. M=0, L=50,
W=80
Cabeçalho (Y/N)? Y
Entre com o cabeçalho da página: DADOS PESSOAIS; ==============
Relatório com espaço duplo (Y/N)? N
Totais requerido (Y/N)? N
Coluna, tamanho, conteúdo
001 20, NOME
Entre com o cabeçalho: NOME; - - - - <ENTER>
002 25, ENDERECO
Entre com o cabeçalho: ENDERECO; - - - - - - - - <ENTER>
003 12, CIDADE
Entre com o cabeçalho: CIDADE; - - - - - - <ENTER>
004 3, ESTADO
Entre com o cabeçalho: ESTADO; - - - - - - <ENTER>
005 5, CEP
Entre com o cabeçalho: CEP; - - - <ENTER>
006 <ENTER>
```

AO TECLAR ENTER NO CAMPO 6 O RELATÓRIO SERÁ MANDADO PARA A TELA AUTOMATICAMENTE.

PARA "CHAMÁ-LO" NOVAMENTE E MANDÁ-LO PARA A IMPRESSORA DIGITE:

```
.REPORT FORM CADASTRO TO PRINT
```

DADOS PESSOAIS
================

NOME	ENDEREÇO	CIDADE	UF	CEP
Cesar Sampaio	Al. do Córrego 23	Goiania	GO	74000
Eduardo Ponte	R. da Paz 851	Vitória	ES	29000
Evaldo Lemos	R. do Ouvidor 275	B. Horizonte	MG	30000
Marcelo Vianna	R. 15 de Novembro 1056	Manaus	AM	69000
Marcio Cunha	R. da Consolação 963	Porto Alegre	RS	90000
Neander Toledo	Av. Paulista 1000	São Paulo	SP	01310
Ricardo Noschese	Av. Ibirapuera 945	São Paulo	SP	01510
Roberto Amorim	Quadra K 931	Brasília	DF	70000
Victor Lima	R. Bela Vista 346	Cuiaba	MT	78000
Wagner Dutra	R. da Liberdade	Blumenau	SC	89100

RESET

O comando RESET é usado para reativar o mapa de bit CP/M após um diskette ter sido trocado. Normalmente, se um diskette é trocado, CP/M não permitirá que nada seja escrito até depois de uma nova partida (warm ou soft boot) tenha sido efetuada.

A emissão de um comando RESET quando não tiver ocorrido nenhuma mudança de disco não tem nenhum efeito.

Este sistema em equipamentos MSX HOTBIT chama-se MCP.

NOTA

Nos Sistemas DOS este comando não tem significado podendo haver trocas de diskettes sem ser necessário usá-lo.

RESTORE

Observações:
RESTORE recupera variáveis de memória de um arquivo de memória.

Sintaxe:
RESTORE FROM (nome do arquivo) {ADDITIVE}

Observações:
O nome do arquivo deve incluir o drive se não for o drive default. A extensão .mem é assumida, a não ser que outra seja especificada.

A menos que a opção ADDITIVE seja usada, todas as variáveis de memória correntes serão regravadas ou apagadas.

Há um limite de 64 variáveis de memória ativas em qualquer momento. 1536 bytes são alocados para as variáveis de memória ativas.

Veja também:
RELEASE, SAVE, STORE

RETURN

RETURN termina um programa e retorna o controle ao programa que o chamou (principal) ou ao modo interativo normal do dBASE II, se o usuário chamou o programa diretamente.

Sintaxe:
RETURN

Observações:
Se o comando RETURN (ou equivalente) não estiver incluído num programa, a marca de fim-de-arquivo do programa terá o mesmo efeito deste comando.

SAVE

SAVE é usado para armazenar todo ou parte do conjunto de variáveis de memória atuais em um arquivo em disco.

Sintaxe:
SAVE TO (nome do arquivo) [ALL LIKE/EXCEPT (máscara)]

Observações:
O nome do arquivo deve incluir a especificação do drive, se não for o drive default.

A menos que outra seja especificada, a extensão para arquivos criados com este comando será .mem.

Na máscara, o ponto de interrogação (?) representa um único caracter, e o asterisco (*) representa um ou mais caracteres. Se o nome do arquivo já existe, ele será regravado se não, será criado.

Se a opção ALL LIKE/EXCEPT não for usada todas as variáveis de memória serão salvas para o arquivo em disco.

Exemplo:
Para salvar todas as variáveis de memória no arquivo Meuarq.mem, no drive B:

```
.SAVE TO B: Meuarq
```

Para salvar todas as variáveis de memória cujos nomes comecem com a letra D para o arquivo em disco Meuarq.mem, no drive B:

```
.SAVE ALL LIKE d* TO B: Meuarq
```

Para salvar todas as variáveis de memória exceto aquelas cujos nomes contenham a letra "x" como quarto caracter, no arquivo Meuarq.ant no drive default:

```
.SAVE ALL EXCEPT ???x*TO Meuarq.ant
```

Veja também:
RESTORE, STORE

SELECT

SELECT (PRIMARY)
(SECONDARY)

Este comando faz com que o dBASE II selecione uma das duas áreas de trabalho possíveis, para operações futuras. Isto permite ao usuário do dBASE II fazer operações nos dois arquivo .DBF ao mesmo tempo, tais como usar os dados de um dos arquivos para atualizar os dados em um outro, ou comparar os dados em dois bancos de dados, ou quaisquer outras multi-operações de banco de dados.

Quando o dBASE II é iniciado, a área PRIMÁRIA está ativa. A área PRIMÁRIA continuará ativa até que uma instrução SELECT SECONDARY seja dada. A área secundária estará então ativa até que um comando SELECT PRIMARY seja encontrada. Um banco de dados diferente pode ser usado em cada uma das áreas. Isto permite o emprego simultâneo de dois bancos de dados de uma vez. Não há nenhum efeito se um comando SELECT SECONDARY é inserido quando a área secundária já está selecionada ou vice-versa com a área primária.

Quando ambas as áreas de trabalho tiverem arquivos de dados em USE, as variáveis de campo podem ser extraídas de qualquer área. Isto equivale dizer que, qualquer expressão pode usar variáveis de ambas as regiões de bancos de dados. Se os nomes dos campos em ambas as regiões são os mesmo para uma variável desejada, então esta variável pode ser pré-fixada pelas letras "P" ou "S" para indicar de que área ela vem.

Comandos dBASE que provocam movimentos do banco de dados (isto é, GOTO, SKIP, REPORT,COPY, LIST, DISPLAY e outros) afetam apenas a área atualmente selecionada. O comando SET LINKAGE ON permitirá à todos os comandos sequenciais (aqueles que tem um parâmetro **<scope>**) executar o posicionamento em ambos os bancos de dados (primário ou secundário). (Veja o comando SET). O comando REPLACE afetará apenas as variáveis no banco de dados atualmente selecionado. O comando DISPLAY STRUCTURE mostrará apenas a estrutura do banco de dados atualmente selecionado.

SET

SET é um comando especial de tela que mostra os parâmetros correntes de processamento e permite selecionar e alterar qualquer parâmetro que possa ser controlado por um comando SET.

Sintaxe:
SET

Observações:
O comando possibilita ver e alterar:
Todos os comandos SET ON/OFF
Atributos da tela
Drive default e caminho de procura de arquivo
Arquivos alternate, de formação e de índice
Relações
Margem

O item e categoria atualmente em vigor são indicados pelo vídeo reverso ou pelo cursor.

SET ALTERNATE

SET ALTERNATE cria um arquivo de texto que pode ser usado para registrar as atividades de processamento de dBASE II.

Sintaxe:
SET ALTERNATE ON/OFF
SET ALTERNATE TO (nome do arquivo)

Observações:
SET ALTERNATE está normalmente em OFF.

Este comando consiste de duas partes:

SET ALTERNATE TO (nome do arquivo) criará um arquivo de texto. O nome do arquivo deve incluir a especificação de drive, se não for o drive default. A extensão do arquivo .txt é assumida a menos que outra seja especificada.

SET ALTERNATE ON irá registrar as entradas de teclado e todas as telas exibidas, no arquivo criado. Operações de tela tais como @...SAY, EDIT e APPEND não são registradas. Este modo de registrar as saídas pode ser ligado ou desligado quantas vezes de desejar.

O arquivo criado por SET ALTERNATE é um arquivo de texto padrão ASCII, e pode ser editado pela maioria dos programas de processamento de texto.

Exemplos:
O arquivo comando abaixo ilustra o comando SET ALTERNATE:

```
SET ALTERNATE TO arqtexto
     (ARQTEXTO.TXT)

SET ALTERNATE ON
     ... comandos

SET ALTERNATE OFF

     ...mais comandos
SET ALTERNATE ON
     ... comandos
SET ALTERNATE OFF
```

SET BELL

SET BELL determina se uma campainha de advertência irá tocar em certas operações.

Sintaxe:
SET BELL ON/OFF

Observações:
SET BELL está normalmente em ON.

Com SET BELL ON, um som de advertência audível é emitido quando o usuário tenta digitar dados inválidos num campo, e também quando um campo está completamente preenchido.

Veja também:
SET CONFIRM

SET CARRY

SET CARRY copia dados do registro anterior para um novo registro quando são usados os comandos APPEND ou INSERT.

Sintaxe:
SET CARRY ON/OFF

Observações:
SET CARRY está normalmente em OFF.

O comando não afeta os comandos INSERT BLANK e APPEND BLANK.

Veja também:
APPEND, INSERT, READ, SET FORMAT

SET CONFIRM

SET CONFIRM é usado para controlar a maneira pela qual o cursor é movimentado de uma variável para outra durante a entrada de dados em tela.

Sintaxe:
SET CONFIRM ON/OFF

Observações:
SET CONFIRM está normalmente em OFF.

No modo normal, o cursor pode avançar de uma variável para a próxima, quando a primeira variável estiver preenchida.

SET CONFIRM ON permite que o cursor avance para o próximo campo somente depois que a tecla <RETURN> for pressionada.

Veja também:
SET BELL

SET CONSOLE

SET CONSOLE liga ou desliga a exibição em vídeo.

Sintaxe:
SET CONSOLE ON/OFF

Observações:
SET CONSOLE está normalmente em ON.

SET CONSOLE afeta somente a saída para o monitor de vídeo. A saída dirigida para a impressora não é afetada.

As entradas pelo teclado serão aceitas, mesmo que não sejam visíveis no monitor de vídeo.

As mensagens de erro não serão omitidas por SET CONSOLE OFF.

SET CONSOLE OFF não deve ser usado em conjunto com o comando DISPLAY.

SET COLON

SET COLON determina como os campos serão mostrados na tela.

Sintaxe:
SET COLON ON/OFF

Observações:
SET COLON está normalmente em ON. Os campos são delimitados pelo ":".

SET COLON OFF elimina os dois pontos (:).

SET COLON TO (string) estabelece os caracteres a serem usados para marcar a área do campo. O string deve ter um ou dois caracteres. Se um caracter é usado este caracter será usado para marcar o início e o fim do campo. Se dois caracteres são usados, o primeiro marcará o início do campo e o segundo marcará o fim do campo. SET COLON deve ser emitido antes dos comandos @ usados com o comando READ.

Exemplos:
Para usar # para marcar o início e o fim de um campo:

```
.SET COLON TO #
.SET COLON ON
```

Para marcar o início de cada campo com [,e o fim com]:

```
.SET COLON TO [ ]
.SET COLON ON
```

Para voltar os delimitadores para dois-pontos:

```
.SET COLON TO DEFAULT
```

> **NOTA**
> Somente a função ON/OFF foi implementada nesta versão.

Veja também:
@, APPEND, CHANGE, EDIT, INSERT, READ

SET DEBUG

SET DEBUG é uma ferramenta para localizar erros num programa.

Sintaxe:
SET DEBUG ON/OFF

Observações:
SET DEBUG está normalmente em OFF.

Quando SET DEBUG está ligado (ON), a saída do SET ECHO é dirigida para a impressora. Isto evita a interferência entre as operações de tela do programa e a exibição em tela, o que normalmente acontece no processo de depuração.

Veja também:
SET ECHO, SET TALK

SET DEFAULT

SET DEFAULT elimina a necessidade de especificar o drive quando estamos trabalhando com arquivos em um drive específico. Todas as operações com arquivo assumirão o drive default, a menos que outro identificador de drive seja especificamente incluído no nome do arquivo.

Sintaxe:
SET DEFAULT TO (drive)

Observações:
Ao entar no dBASE II, o drive default é aquele que contém o programa DBASE. COM.

Exemplos:
O arquivo Cadastro está no drive B.
O arquivo dBase. COM está no drive A. O drive default é o A:

```
.USE Cadastro
Arquivo não existe
```

(Cadastro não está no drive default A.)

```
.SET DEFAULT TO B
.USE Cadastro
```

(Cadastro está no drive default B.)

SET DELETED

SET DELETED determina se os registros que tenham sido marcados para eliminação deverão ser incluídos ou ignorados pelos outros comandos do dBASE II.

Sintaxe:
SET DELETED ON/OFF

Observações:
SET DELETED está normalmente em OFF.

INDEX e REINDEX sempre incluem todos os registros sem levar em conta o status do SET DELETED.

Quando o SET DELETED está ON, os registros deletados são ignorados por todos os comandos.

SET ECHO

SET ECHO mostra as linhas de comando dos programas do dBASE II na tela e/ou impressora durante a execução do programa. O comando é basicamente uma ferramenta de depuração.

Sintaxe:
SET ECHO ON/OFF

Observações:
SET ECHO está normalmente em OFF.

Ilustrações do programa não são normalmente mostradas durante a execução.

SET ECHO é uma das quatro ferramentas de depuração do dBASE II outras três são: SET DEBUG, SET STEP, e SET TALK.

SET ECHO pode ser usado em conjunto com SET DEBUG para evitar que a tela venha a tornar-se confusa.

Veja também:
SET DEBUG, SET STEP, SET TALK

SET EJECT ON/OFF

SET EJECT faz com que a impressora avance o papel para o início para a próxima folha.

Sintaxe:
SET EJECT ON/OFF

Observações:
SET EJECT emite um caracter de alimentação da folha para a impressora. Para uma operação satisfatória da impressora, o papel deve ser posicionado no topo da folha. Consulte o manual da impressora para instruções.

A ativação da impressora quando ele não estiver conectada ou ligada poderá travar o computador. Conectando e ligando a impressora, o computador deverá destravar. Se não, o computador deverá ser reiniciado. Consulte o manual de operação do computador para instruções.

SET ESCAPE

SET ESCAPE determina se um programa ou a execução de um comando pode ou não ser abortado pela tecla Esc.

Sintaxe:
SET ESCAPE ON/OFF

Observações:
SET ESCAPE está normalmente em ON. Isto significa que a tecla Esc está ativa.

Quando a tecla Esc está ativa, (ON), e é pressionada durante a execução de um programa, o dBASE II mostrará a mensagem ***INTERROMPIDO*** e voltará para o prompt de ponto.

SET EXACT

SET EXACT determina como uma comparação entre dois strings será avaliada. Isso irá afetar o resultado da avaliação de uma condição.

Sintaxe:
SET EXACT ON/OFF

Observações:
SET EXACT está normalmente em OFF.

Comparação entre strings de caracter começaram com o caracter mais à esquerda de cada string, e continuam caracter por caracter até o fim do menor string. Brancos excedentes são descartados para o propósito de avaliação.

Exemplos:
Considere as duas strings de caracter ABC e ABCDEF.

Primeiro, compare as duas strings de caracter com SET EXACT OFF.

```
.?ABC =ABCDEF
.F.
.?ABCDEF = ABC
.T.
```

Se o string à direita do sinal for igual aos primeiros caracteres do string maior, à esquerda do sinal, os dois strings serão considerados iguais.

Com SET EXACT ON, a última comparação não é mais válida:

```
.? ABCDEF=ABC
.F.
```

Os dois strings não serão mais considerados iguais; não há uma correspondência exata.

SET FORMAT

SET FORMAT permite que o usuário selecione formatos especiais, armazenando como arquivos de formatação (fmt.)

Sintaxe:
SET FORMAT TO (nome do arquivo)

Observações:
Se o comando não é usado, são assumidos os padrões do APPEND, CHANGE, EDIT e INSERT.

Se um nome de arquivo de formatação é usado sem extensão, é assumida a extensão .fmt.

Exemplos:
Para usar a tela Exemplo.fmt com o arquivo de dados ativo:

```
.SET FORMAT TO Exemplo
```

Veja também:
APPEND, CHANGE, EDIT, INSERT, READ

SET FORMAT TO

SET FORMAT TO determina se os comandos @ serão direcionados para o monitor de vídeo ou para a impressora.

Sintaxe:
SET FORMAT TO <PRINTER/SCREEN>

Observações:
O default é SET FORMAT TO SCREEN.

Se o SET FORMAT TO PRINTER é usado, os comandos @...GET serão ignorados e o comando @ SAY serão enviados para a impressora.

Exemplos:
Usando SET FORMAT para imprimir um arquivo de formatação denominado W2:

```
.SET FORMAT TO PRINTER
.SET FORMAT TO W2
.READ
```

Veja também:
@, SET FORMAT

SET HEADING TO

SET HEADING TO oferece a condição de ser colocado o cabeçalho, caso este tenha sido negado ao se entrar com as opções do Report Form.

Sintaxe:
SET HEADING TO <String>

SET INDEX

SET INDEX abre os arquivos índice nomeados no comando.

Sintaxe:
SET INDEX TO [**<lista de nomes de arquivos índice>**]

Observações:
A extensão .ndx será assumida a menos que outra seja especificada.

Até sete arquivos índice podem ser selecionados com SET INDEX. O primeiro da lista é o arquivo de controle.

Entretanto, todos os arquivos de índice ativos são automaticamente atualizados sempre que ocorre uma alteração no arquivo de dados associado.

O ponteiro de registros é posicionado no início do arquivo de acordo com o arquivo índice de controle.

O comando INSERT é equivalente ao APPEND quando um ou mais arquivos índice estão em uso.

Exemplos:
Um arquivo de dados Alunos contém os Nomes e Salas dos estudantes. O arquivo está indexado pelo nome dos estudantes no arquivo índice Nomes, e por sala e nome no arquivo índice Classe. Para usar o banco de dados com os nomes em ordem alfabética.

```
.USE Alunos INDEX Nomes Classe
```

Para re-selecionar os índices para que Classe seja o arquivo índice de controle:

```
.SET INDEX TO Classe Nomes
```

Veja também:
INDEX, REINDEX, USE

SET LINKAGE

Todos os comando sequênciais (LIST, REPORT, SUM), aqueles que permitem que um escopo incremente os indicadores de registros do PRIMÁRIO e SECUNDÁRIO simultâneamente, mas não os força a serem iguais. Quando está em SET ON, uma ligação é estabelecida entre os arquivos dos primário e do secundário. A ligação é baseada na posição lógica do arquivo. Note que esta posição é a mesma que a posição física a menos que um arquivo de índice esteja em uso. Quando este é usado, a posição lógica do arquivo é a posição do arquivo de índice. Uma vez que dois bancos de dados estão ligados, os indicadores de registros em ambos os arquivos movem-se juntos quando qualquer comandos do dBASE II que tem um escopo como parte de sua sintaxe é colocado. Para usar o comando, abra um banco de dados em ambas as áreas de trabalho primária e secundária e coloque o comando SET LINKAGE ON. Os comandos para executar esta tarefa podem ser colocados em qualquer ordem. Por exemplo, a seguinte sequência de comando:

```
CLEAR
SET LINKAGE ON
USE UM
SELECT SECONDARY
USE DOIS
SELECT PRIMARY
```

Depois que estas sequências de comandos forem entradas, a ligação entre os arquivos UM e DOIS é ativada. Neste ponto, um comando DISPLAY, LIST ou REPORT pode ser usado para visualizar informação de ambos os arquivos. Assumindo que o arquivo UM tem um campo chamado Fnome e que o arquivo DOIS tem um campo chamado Lnome, o seguinte comando mostraria estes campos de ambos os arquivos, avançando o indicador de registro em ambos:

```
.LIST Fnome, S.Lnome
```

Note a inclusão do prefixo "S." com o campo Lnome. Este prefixo é usado neste exemplo porque a área de trabalho primária está ativada. Prefixos para campos em áreas de trabalho não selecionados devem ser usados em todos os comandos com SET LINKAGE.

Se em ambos os arquivo um têm mais registros que outro, os registros finais no arquivo mais comprido são ignorados, enquanto SET LINKAGE está ON. Se um dos arquivos está indexado, o indicador de registros move de acordo com a ordem do arquivo de índice. Note que os comandos GO/GOTO e SKIP não causam um movimento do indicador de registro em um arquivo não selecionado porque estes dois comandos não tem escopo. No mais, se um comando que move o arquivo não selecionado para o seu final é colocado enquanto SET LINKAGE está ON, este arquivo deve ser reposicionado antes de qualquer outro comando. Como um exemplo, suponha que a ligação tenha sido ativada e o comando LIST apenas mostrou o que foi colocado. O arquivo SECUNDÁRIO está no fim do arquivo. Para LISTAR todos os registros de ambos os arquivos novamente, deverá ser usado o comando "GOTO TOP" selecionando ambos os arquivos e repetindo-se a sequência da página anterior.

SET INTENSITY

SET INTENSITY determina se os campos serão mostrados em vídeo reverso nas operações de tela tais como o EDIT e APPEND.

Sintaxe:
SET INTENSITY ON/OFF

Observações:
SET INTENSITY está normalmente em ON (isto é, o vídeo reverso está ativo).

Esse comando comuta entre dois conjuntos de atributos de tela: padrão e realçado.

Quando o INTENSITY é ativado pelo ON, os atributos padrão e realçados podem ser diferentes (como no caso default).

Quando INTENSITY está OFF, o atributo realçado é usado tanto pelo modo padrão como pelo modo realçado.

NOTA:

Nesta versão este comando não foi implementado, ou seja, o comando é tratado sempre como SET INTENSITY OFF.

SET MARGIN

SET MARGIN ajusta a margem esquerda para as saídas em impressora. A exibição em vídeo não é afetada.

Sintaxe:
SET MARGIN TO <expN>

Observações:
O valor default da margem esquerda é zero (0).

Exemplos:
Para colocar a margem esquerda a 10 espaços à direita da primeira posição da impressão:

```
.SET MARGIN TO 10
```

SET PRINT

SET PRINT ON direciona todas as saídas não formatadas pelo comando @...SAY, para a impressora e para a tela.

Sintaxe:
SET PRINT ON/OFF

Observações:
SET PRINT está normalmente em OFF.
Ativar a impressora quando esta não está conectada ou ligada pode travar o computador. Conectando e ligando a impressora, o computador deve destravar. Se não, o computador deve ser reinicializado. Consulte o manual de operações do computador para instruções.

SET STEP

SET STEP detém a execução de um programa após cada instrução. Isto permite ao programador seguir o programa passo a passo. O comando é basicamente uma ferramenta de depuração.

Sintaxe:
SET STEP ON/OFF

Observações:
SET STEP está normalmente em OFF.
SET STEP é usado normalmente como uma ferramenta de depuração.

Durante a operação com SET STEP ON, o programa executa apenas uma instrução de cada vez. O resultado de cada operação é mostrado, e em seguida, antes da próxima instrução ser executada, aparecerá esta mensagem:

```
Pressione qualquer tecla para execução - Esc para cancelar
```

O processamento ficará parado até que o usuário entre com uma das duas opções.

Veja também:
SET DEBUG, SET ECHO, SET TALK

SET TALK

SET TALK determina se o resultado de muitos comandos do dBASE II será ou não mostrado.

Sintaxe:
SET TALK ON/OFF

Observações:
SET TALK está normalmente em ON.

SET TALK ON é o modo conversacional, que fornece uma resposta visual para quase todos os comandos. Está usualmente ativo durante o uso interativo.

SET TALK OFF é usado em programas para controlar o que aparece na tela (e na impressora).

Veja também:
SET DEBUG, SET ECHO

SKIP

SKIP move o ponteiro de registro para diante ou para trás no arquivo de dados ativo.

Sintaxe:
SKIP [+ / – <expN>]

Observações:
Se a expressão não for incluída, o dBASE II avançará um registro.

Se ambos os sinais (+ / –) forem omitidos, o ponteiro de registro avança.

Se um arquivo índice está em uso com o arquivo de dados, SKIP segue a ordem do índice.

Se um SKIP for emitido quando o ponteiro de registro está no último registro do arquivo, o valor de + / – for maior do que o número do último registro no arquivo EOF será setado para Verdadeiro.

Exemplo:
O arquivo de dados Cadastro foi usado para os seguintes exemplos:

Para avançar 1 registro no arquivo de dados:

```
.USE Cadastro
.SKIP
RECORD: 00002
```

Para avançar 5 registros:

```
.STORE 5 TO PULA
.SKIP PULA
RECORD 00007
```

Para retroceder 3 registros:

```
.SKIP-3
RECORD: 00004
```

STORE

STORE cria e inicializa uma ou mais variáveis de memória.

Sintaxe:
STORE **<expressão>** TO **<lista de variáveis de memória>**

Se a variável de memória já existe, será regravada.

Uma única expressão pode ser armazenada em várias variáveis de memória com um único STORE.

Uma variável numérica pode ser adicionada a si mesma; uma variável do tipo caracter pode ser concatenada a si mesma.

Um campo e uma variável podem ter o mesmo nome. O campo terá prioridade em todas as operações subsequentes.

Exemplos:
Para armazenar um falso lógico na variável de memória Verdade:

```
.STORE .F. TO Verdade
```

Para armazenar o número 0 nas variáveis de memória A, B e C:

```
.STORE 0 TO A,B,C
```

Para armazenar a expressão Custo * 0,06 em taxa:

```
.STORE CUSTO * 0,06 TO taxa
```

Para armazenar o conteúdo do campo Grau na variável de memória Grau:

```
.STORE Grau TO Grau
```

Para ver o campo Grau:

```
.? Grau
```

SUM

SUM totaliza expressões envolvendo campos numéricos em todos os registros, ou num subconjunto designado do arquivo de dados ativo.

Sintaxe:
SUM [(escopo)] [(lista de expressões)] [TO (lista varmem)] [FOR /WHILE (condição)]

Observações:
A menos que de outro modo seja especificado por um escopo ou cláusula FOR /WHILE, todos os registros serão somados.

Se nenhuma lista de expressões estiver presente, todos os campos numéricos serão somados.

TEXT

TEXT é usado em programas para enviar blocos de texto para a tecla ou para impressora. O comando fornece uma maneira simples e conveniente de escrever em um periférico de saída.

Sintaxe:
TEXT
(caracteres do texto)
ENDTEXT

Observações:
O texto será enviado exatamente como foi originalmente digitado no programa.

A função macro (&) não será interpretada quando embutida no texto. Ela será impressa como um E comercial (&).

Exemplos:
Um exemplo do comando TEXT é mostrado abaixo:

```
TEXT

Este é um exemplo de TEXTO a ser exibido pelo comando
TEXT. Este texto será enviado diretamente para a tela sem ser
interpretado pelo dBASE II.
ENDTEXT
```

TOTAL

TOTAL soma os campos numéricos do arquivo ativo e registra os resultados num segundo arquivo de dados. Os campos numéricos no arquivo de destino conterão os totais de todos os registros sequênciais que tiverem o mesmo valor de chave no arquivo de origem.

Sintaxe:
TOTAL TO (campo chave) TO (nome do arquivo) [(escopo)]
[FIELDS (lista de campos)]
[FOR/WHILE (condição)]

Observações:
O nome do arquivo deve incluir o drive, se não for o drive default.

Uma extensão .dbf será assumida a menos que outra seja especificada.

Se o nome do arquivo destino já existir, o comando total vai regravar o arquivo. Se não, o arquivo será criado. Em ambos os casos, a estrutura do arquivo será idêntica à do arquivo ativo. Campos do tipo memo não podem ser incluídos no arquivo de destino.

A menos que de outro modo especificado pelo escopo ou pela cláusula FOR/WHILE, todos os registros serão totalizados.

A menos que de outro modo especificado pela cláusula FIELDS, todos os campos numéricos do arquivo serão totalizados.

O arquivo ativo deve estar indexado ou classificado pelo campo-chave.

Todos os registros com o mesmo valor no campo-chave especificado serão totalizados em um único registro no arquivo de destino.

Todos os campos numéricos especificados na lista de campos conterão totais. Todos os outros registros conterão os dados do primeiro registro do conjunto dos campos-chave.

Se o tamanho do campo não for suficiente para acomodar o total, o dBASE II responde com uma mensagem de erro e coloca asteriscos no campo.

Exemplos:
O banco de dados, classificado ESTOQUE será totalizado por item no arquivo RESUMO.

```
.USE Estoque
.TOTAL ON item TO resumo
```

ESTOQUE

ITEM	QTD
prego	35
prego	27
tacha	125
parafuso	10
parafuso	20
parafuso	5

RESUMO

Item	QTD
prego	62
tacha	125
parafuso	35

UPDATE

UPDATE usa dados de um arquivo de origem para fazer alterações nos registros do arquivo ativo.

As alterações são efetuadas pelo "casamento" dos registros nos dois arquivos de acordo com um único campo chave.

Sintaxe:
UPDATE FROM <file> ON **<chave>**
[ADD **<FIELD LIST>**]
[RANDOM] [REPLACE [**<field list>**]]
[,field> **WITH <from-field>**]

Observações:
O arquivo de dados a ser atualizado deve ser o arquivo ativo na área de trabalho atualmente selecionada; o arquivo de origem deve estar ativo um uma das área de trabalho não selecionadas.

O campo-chave deve ser o mesmo (com o mesmo nome) em ambos os arquivos.

Ambos os arquivos devem estar indexados ou classificados pelo campo chave, a não ser que a cláusula RANDOM seja usada. RANDOM requer que o arquivo-destino esteja indexado pelo campo chave. O arquivo-origem pode estar em qualquer ordem.

Se o campo-chave no destino não é único para cada registro de destino, somente o primeiro registro de cada conjunto será atualizado.

Exemplos:
A ilustração adiante descreve a atualização do arquivo-destino. Estoque a partir do arquivo-origem Fatura. O campo chave que relaciona estes dois arquivos é item. Ambos os arquivos estão classificados, de modo que não se usou o RANDOM. O conteúdo do campo de origem Qtd será somado ao campo de destino Qtd, para todos os itens que casem. O arquivo ativo é Estoque.

```
.SELECT SECONDARY
.USE Fatura
.SELECT PRIMARY
.USE Estoque
.UPDATE ON Item FROM Fatura REPLACE Qtd WITH Qtd + Fatura
```

ESTOQUE

ITEM	QTD
MARTELO	3
ALICATE	2
ROLDANA	1
LIXA	10
SOQUETE	5

+

FATURA

ITEM	QTD
MARTELO	2
ROLDANA	2
LIXA	10

=

ESTOQUE

ITEM	QTD
MARTELO	5
ALICATE	2
ROLDANA	3
LIXA	20
SOQUETE	5

USE

USE abre ou fecha um arquivo de dados existente e até sete arquivos de índice na área de trabalho atualmente selecionada.

Sintaxe:
USE (nome arq.) [INDEX (lista arq índice)]

Observações:
Se USE for emitido sem qualquer parâmetro, o arquivo de dados ativo e os arquivos de índice na área de trabalho atualmente selecionada serão fechados.

A menos que outra seja especificada, o dBASE II assume a extensão .dbf para o nome do arquivo e extensões .ndx para os arquivos de índice.

Quando um arquivo de dados é usado sem qualquer arquivo de índice, o ponteiro de registro é posicionado no primeiro registro do arquivo.

Quando um arquivo de dados é usado com um ou mais arquivos de índice, o ponteiro de registro é posicionado no primeiro registro lógico de índice.

Exemplos:
O banco de dados denominado Cadastro possui arquivos de índice Cadpess e Cadcid.

```
.USE Cadastro
```

Para abrir este banco de dados com seus dois arquivo de índice:

```
.USE Cadastro INDEX Cadpess, Cadcid
```

WAIT

WAIT faz com que o processamento do dBASE II pare até que qualquer tecla seja pressionada.

Sintaxe:
WAIT [TO (varmem)]

Observações:
Se a opção (varmem) for incluída, a entrada do teclado é armazenada na variável de memória indicada.

A variável de memória criada será do tipo caracter. Se um (RETURN) ou outro caracter sem correspondente impresso for digitada, o conteúdo da variável será nulo (ASCII).

A mensagem que aparecerá na tela será: ESPERANDO.

FUNÇÕES

As funções realizam operações específicas que aumentam e melhoram o desempenho dos comandos dBASE II.

As funções são sempre usadas como expressões, ou em expressões que seguem um verbo do comando. As funções e os tipos de dados que elas produzem estão listados em ordem alfabética na tabela abaixo:

Nome da Função	Descrição	Saída Tipo/Dado	Entrada Tipo/Dado
&	Macro Substituição	C	C
@	Pesquisa de Substring	C	C
RANK	Conversão de Caracter para código ASCII	N	C
CHR	Conversão de código para caracter	C	C
DATE	Data do sistema	D	*
*	Registro deletado	L	*
EOF	Fim de arquivo	L	*
FILE	Existência de arquivo	C	C
INT	Inteiro	N	N
LEN	Tamanho do string de caracter	N	C
#	Número de registro atual	N	*
STR	Conversão de número para caracter	C	N
$	Seleção de substring	C	C
TRIM	Remoção de espaços em branco	C	C
TEST	Validação de expressão	C	C
!	Conversão de minúscula para maiúscula	C	C
VAL	Conversão de caracter para número	N	C
TYPE	Tipo de variável	C	C

Observação:
O * indica que a entrada deste tipo de dados não é aplicável.

As funções estão por sua classificação funcional na tabela abaixo:

Classificação funcional	Nome da Função	Descrição
DATA	DATE	Data do sistema
MANIPULAÇÃO DE CARACTERES	&	Macro substituição
	@	Pesquisa de substring
	$	Seleção de substring
	Trim	Remove espaços em branco
	!	Produz maiúsculas
MATEMÁTICA	INT	Inteiro
CONVERSÃO	RANK	Caracter para código ASCII
	CHR	Código ASCII para caracter
	STR	Numérico para caracter
	!	Minúscula para maiúscula
	VAL	Caracter para numérico
TESTES ESPECIALIZADOS	*	Registro deletado
	EOF	Fim de arquivo
	FILE	Existência de arquivo
	LEN	Tamanho do string
	#	Número do registro atual
	TEST	Validação da expressão
	TYPE	Tipo de caracter

& - FUNÇÃO MACRO

& é Função Macro. É usada para substituir o conteúdo de uma variável de memória nos casos em que o dBASE II consideraria o nome da variável literalmente. Esta função somente pode ser usada para variáveis do tipo caracter.

Sintaxe:
& <variável caracter>

Exemplos:
Para usar o comando FIND com uma variável de memória:

```
.STORE "datalogica" TO Mnome
datalogica
.FIND &Mnome
```

O dBASE II normalmente consideraria literalmente a sequência de caracteres após o FIND. Sem o &, o dBASE II iria procurar no banco de dados a sequência de caracteres Mnome, não datalogica. Para substituir o conteúdo de um nome de variável de memória num string:

```
.Store "datalogica" TO Nome
datalogica
.STORE "Ola &nome" TO frase
Ola datalogica
.? Frase
Ola datalogica
```

Para inserir o conteúdo de uma variável de memória numa sequência de caracteres que não contém brancos:

```
.store "x" TO alfa
x
.store "12&ALFA.34" To Mostra
.?Mostra
12x34
```

Quando caracteres não-brancos são inseridos no conteúdo de uma variável de memória, como no terceiro exemplo, usa-se um ponto para indicar o fim do nome da variável.

FUNÇÃO RANK

Rank retorna o valor em código ASCII do primeiro caracter à esquerda num string:

Sintaxe:
RANK ("<string>")

Exemplos:

```
.? RANK ("Nestle")
```

(O código ASCII para N é 78)

```
.STORE "123" TO Número
123
.?RANK (NÚMERO)
31
```

(O código ASCII para 1 é 31)

FUNÇÃO @

@ é função de pesquisa de substring. Ela retorna um número que mostra a posição inicial de um string dentro de um segundo string de caracter. O string contido é chamado substring. Se o substring não estiver contido dentro do string, a função retornará zero (0).

Sintaxe:
@ (" <string 1> ", " <string 2> ")

Exemplos:

```
.? @ ("Silva", "Ana Silva")
5
```

O substring "Silva" começa no quinto caracter do string "Ana Silva".

```
.STORE "Ana Silva" TO Alfa
Ana Silva
.STORE "na" TO Beta
na
.? @ (Beta, Alfa)
2
```

FUNÇÃO CHR

CHR é a função número-para-caracter. Esta função permite usar os efeitos especiais disponíveis na impressora e/ou na tela. CHR é usado para transferir dados tipo caracter por meio de valores do código ASCII para os caracteres de teclado, assim como enviar caracteres para os quais não há teclas equivalentes.

Sintaxe:
CHR (<n>)

Observações:
<n> é um número inteiro de 0 a 255.

Exemplos:
Para mostrar o caracter ASCII cujo valor é 65.

```
.?CHR(65)
A
```

O código ASCII para A maiúsculo é 65.)

Na maioria dos sistemas aciona-se a campainha por meio do código ASCII 7. Para soar a campainha:

```
.?CHR(7)
```

Para soar a campainha e escrever um texto na tela:

```
.? CHR(7) + "Seja mais cuidadoso"
Seja mais cuidadoso
```

Muitos dispositivos usam os códigos ASCII para mostrar caracteres. CHR fornece o recurso para usar os caracteres para os quais não existem teclas equivalentes. Assim, CHR (22) mostrará uma barra vertical.

```
.?CHR(22)
|
```

FUNÇÃO DATE

Date retorna a data do sistema na forma dd/mm/aa.

Sintaxe:
Date( )

Exemplos:
Para mostrar a data do sistema:

```
.?DATE( )
05/04/84
```

Para armazenar a data do sistema numa variável de memória Mdata:

```
.STORE DATE( ) TO MDATA
05/04/84
```

FUNÇÃO *

* (DELETED) identifica os registros que foram marcados para deleção. Um true (.T.) lógico retorna se o registro atual foi marcado para deleção.

Sintaxe:
*

Exemplos:
Para mostrar todos os registros do arquivo de dados ativo que foram marcados para deleção:

```
.DISPLAY FOR *
```

Para determinar se o registro atual foi marcado para deleção:

```
.? *
.T.
```

(O registro está marcado para deleção.)

```
.? *
.F.
```

(O registro não está marcado para deleção.)

Observação:
NOTE QUE O ASTERISCO SERVE TANTO PARA A FUNÇÃO DELETED QUANTO PARA COMENTÁRIOS DENTRO DE PROGRAMAS.

Veja também:
SET DELETED ON

FUNÇÃO EOF

EOF (End of file) indica o fim de um arquivo. Um True lógico (.T.) retorna quando o último registro lógico de um arquivo em uso é ultrapassado. (o flag fim-de-arquivo é setado).

Sintaxe:
EOF

Exemplos:
Para testar o fim-de-arquivo:

```
.GOTO TOP
.? EOF
.T.
.GOTO BOTTOM
.? EOF
.F.
```

(O ponteiro do registros está no fim-de-arquivo.)

(O ponteiro do registro não está no fim-de-arquivo.)

FUNÇÃO FILE

FILE verifica a existência de um nome de arquivo específico.

Um valor lógico True (.T.) retorna se o arquivo existe.

Sintaxe:
FILE (<**nome do arquivo**>)

Observações:
A função FILE vai sempre retornar um valor lógico False (.F.) para um arquivo aberto.

Exemplos:
O arquivo Cadastro .dbf está no drive B, o drive corrente é o A e o drive default também é o A.

```
.? FILE ("Exemplo. dbf")
.F.
.? FILE ("B: EXEMPLO .dbf")
.T.
.? FILE ("B: Exemplos")
.F.
.STORE "B:Exemplo .dbf" TO teste
B:Exemplos.dbf
.FILE (&teste)
.T.
```

(O arquivo não está no drive A.)

(Está faltando a extensão do arquivo.)

FUNÇÃO INT

INT é a função inteiro. INT converte uma expressão numérica em número inteiro, desprezando todos os dígitos à direita do ponto decimal.

Sintaxe:
INT (<expN>)

Exemplos:
Para converter o número 10.23 em número inteiro:

```
.? INT (10.23)
10
.STORE 10.23 TO X
10.23
.? int (X)
10
.STORE INT (10.23) TO X
10
```

FUNÇÃO LEN

LEN é a função comprimento. Ela retorna um valor numérico que é o número de caracteres existentes em um string.

Sintaxe:
LEN (<string>)

Exemplos:
Para determinar o número de caracteres no string "Ana Silva":

```
.? LEN ("Ana Silva")
9
```

```
.STORE "Ana Silva" TO Nome
Ana Silva
.? LEN (Nome)
9
```

Se um arquivo de dados contém um campo do tipo caracter chamado Endereço com tamanho de 30 posições, determinar o número de caracteres armazenados no campo corrente cujo conteúdo é:
(Av. Paulista, 2028 - cap.)

```
.?LEN (Endereço)
30
.? LEN (TRIM(Endereço))
24
```

(Este é o tamanho exato do campo.)

(Este é o tamanho do endereço atual.)

FUNÇÃO

é a função registro atual. Ela permite acessar o número do registro atual do arquivo selecionado.

Sintaxe:
#

Observações:
Se não houver registros no arquivo de dados, esta função indicará registro 1 e o flag fim-de-arquivo (EOF) será setado.

Exemplos:
Para mostrar o número do registro atual:

```
.?#
7
```

(O registro atual é o registro número 7.)

Para armazenar o número do registro atual em uma variável de memória:

```
.STORE # TO Alpha
.? Alpha
7
```

FUNÇÃO STR

Observações:
STR converte uma expressão numérica em string.

Sintaxe:
STR (**<expressão numérica>**)
[, **<comprimento>**,] [, **<decimais>**,]

Observações:
O comprimento é o número total de posições que será gerado pelo STR, incluindo o ponto decimal (se houver) e o número total de casas decimais (se houver).

Decimais é o número total de casas decimais a ser gerado.

Se o comprimento do string de saída for menor do que o número de dígitos do número, a saída consistirá de asteriscos.

Se não forem especificadas casas decimais, o número será tratado como um inteiro, e as casas decimais (se houver) serão desprezadas.

Exemplos:
Para mostrar o número 11.14 * 10 como um string:

```
.?STR (11.14,5,2) + "*" + STR (10)
11.14*10
```

FUNÇÃO $ (SUBSTRING)

$ é a função substring. Ela extrai uma parte específica de um string de caracter.

Sintaxe:
$ (<string>, <posição inicial> [, **<número de caracteres>**])

Observações:
Se o número de caracteres é omitido, o substring inicia na posição inicial e termina no último caracter do string original.

Se o número de caracteres informado é maior que o número de caracteres entre a posição de início e do fim do string original, o substring de saída será igual ao substring começando a partir da posição de início e terminando no último caracter do string original.

Se a posição de início especificada for maior que o comprimento do string um string nulo será a saída.

Exemplos:
Para extrair o substring "59" do string "1958 1959 1960":

```
.? $ ("1958 1959 1960",8,2)
   59
```

```
.STORE 'Ingles Espanhol Italiano Alemao Frances' TO Linguas
Ingles Espanhol Italiano Alemao Frances
.? $ (Linguas,8,8)
Espanhol

.STORE $ (Linguas,17,8) TO L1
Italiano
.?L1
Italiano
```

FUNÇÃO TRIM

TRIM remove os espaços em branco do final de um string.

Sintaxe:
TRIM ('<string>')

Exemplos:
Assumindo que um arquivo de dados contém os campos Nome e Endereço.

```
.USE Cadastro
.GOTO RECORD 2
.? Nome, Endereço
Victor Lima                    R. Bela Vista 346
```

O endereço está separado do nome por um número de espaços não utilizados no campo Nome.

Para eliminar os espaços em branco entre o nome e o endereço:

```
.? TRIM (Nome), Endereço
Victor Lima R. Bela Vista 346
```

O espaço que agora separa o Nome do Endereço é causado pela vírgula no comando.

FUNÇÃO TYPE

TYPE avalia uma expressão e retorna um único código de caracter que indica se a expressão é do tipo caracter, numérico, lógico, ou indefinida. É um teste para determinar a existência de uma variável e/ou a validade de uma expressão.

Sintaxe:
TYPE (**<expressão>/variável**)

Observações:
Todas as expressões deverão estar entre aspas simples ou duplas, ou entre colchetes ou parênteses. A omissão resultará em erro de sintaxe se a variável de memória não existe, ou provavelmente em resposta errada U (Undefined, isto é, Indefinida), se a variável existe.

Exemplos:
Para determinar se a variável de memória Teste existe:

```
.? TYPE("Teste")
C
? TYPE(Teste)
U
.STORE "testando 1 2 3" TO Teste
testando 1 2 3
.? TYPE (Teste)
C
```

Para avaliar a expressão, Teste/100, e determinar se é válida (este teste pode ser utilizado para evitar erros de sintaxe):

```
.STORE "ola" TO Teste
ola
.? TYPE ("Teste"/100)
*** erro de sintaxe ***
.STORE 8 TO Y
8
.? TYPE(Y/4)
N
```

Quando os delimitadores são omitidos, TYPE vai avaliar uma variável que é o conteúdo da variável nomeada.

```
.STORE 5 TO Num
5
.STORE NUM 'TO Exemplo
.? TYPE (Exemplo)
N
```

(Num está contida em Exemplo e é uma variável numérica.)

FUNÇÃO !

! é a função de letras maiúsculas. Ela converte caracteres minúsculos em caracteres maiúsculos.

Sintaxe:
! (string de caracter)

Exemplos:

```
.? ! (Este e um belo dia)
     ESTE É UM BELO DIA
```

```
.STORE "Este e um belo dia" TO Belodia
.? Belodia
Este e um belo dia
```

Para exibir esta variável em maiúsculas:

```
.? ! (Belo dia)
BELO DIA
```

```
.? Belodia = "ESTE E UM BELO DIA"
.F.
.! (Belodia) = "ESTE E UM BELO DIA"
.T.
```

FUNÇÃO VAL

VAL é a função de conversão caractere-para-numérico. VAL permite que operações aritméticas sejam realizadas com dados do tipo caracter.

Sintaxe:
VAL ('string de caracter')

Observações:
O valor de todos os caracteres não-numéricos é zero.

Deve-se tomar cuidado quando uma sequência de caracteres numéricos contém um ponto decimal. Parecerá que VAL descarta o ponto decimal e todos os dígitos subsequentes, porém estes dígitos não foram descartados. Eles podem ser vistos com a função STR.

Exemplos:
O valor de um caracter não-numérico é zero.

```
.? VAL("ABC")
    0
```

```
.? VAL("123.45")
123
.STORE "123.45" TO X
123.45
.STORE VAL(X) TO Y
123
.? Y
123
.? STR(Y,6,2)
123.45
.? 0.00 + Y
123.45
```

(123.45 é um string de caracter.)

(Y é um número.)

COMANDO PLUS

Introdução

Você verá agora o mais novo comando do dBASE II: o comando PLUS, que juntamente com suas rotinas apresentadas nesta parte do manual lhe trará um novo dimensionamento no tratamento de seu banco de dados.

BUFF

Comando que aciona/desaciona o buffer de teclado.

Sintaxe:
PLUS BUFF

CDATE

Comando que determina a validade de uma data.

Sintaxe:

	<varmem>	
PLUS CDATE	DATA	TO RESP
	<ddmmaa>	(1 posição)

Onde:

DATA: Variável tipo caracter que deve conter a data a ser consistida com tamanho de 6 posições.

RESP: Variável tipo caracter, tamanho de 1 posição. Retorna "1" caso seja Data Válida, caso contrário "0".

Exemplo:

```
STORE "190361" TO DATNASC
STORE" " TO RESP
PLUS CDATE DATNASC TO RESP
IF RESP = "1"
? "DATA DE NASCIMENTO OK"
ELSE
? "DATA DE NASCIMENTO INVÁLIDA"
ENDIF
DATA DE NASCIMENTO OK
```

DDATE

Comando que calcula a quantidade de dias existentes entre duas datas.

Sintaxe:

	<varmem>	<varmem>	
PLUS DDATE	DAT1	DAT2	TO DIAS
	<ddmmaa>	<ddmmaa>	(5 posições)

Onde:

DAT1: Variável caracter com 6 posições que deve conter a data inferior.

DAT2: Variável caracter com 6 posições que deve conter a data posterior.

DIAS: Variável caracter com 5 posições, que retornará com número de dias existentes entre as datas.

Exemplo:

```
STORE "010485" TO DATANT
STORE "110485" TO DATATO
STORE "        " TO DIAS
PLUS DDATE DATANT DATATO TO DIAS
? "A QUANTIDADE DE DIAS É: " + DIAS
A QUANTIDADE DE DIAS É 00011
```

ADATE

Comando que acrescenta dias em uma data.

Sintaxe:
PLUS ADATE DATA1 DIAS PAR TO DATA2

Onde:

DATA1: Variável tipo caracter com 6 posições que contém a data inicial.

DIAS: Variável caracter com 3 posições que deve conter a quantidade de dias que serão somados em DATA1.

PAR: Variável tipo caracter com 1 posição. Se igual a "0" somará os dias a partir da data informada (DATA1).

Se igual a "1" somará os dias a partir do primeiro dia do mês seguinte a data informada (DATA1).

DATA2: Variável caracter com 6 posições que retornará com a nova data.

Exemplo:

```
STORE "190385" TO DATA1
STORE "002" TO DIAS
STORE "0" TO PAR
STORE "       " TO DATA2
PLUS ADATE DATA1 DIAS PAR TO DATA2
? "A NOVA DATA E:" + DATA 2
A NOVA DATA E: 210385.
```

TDATE

Comando que acrescenta dias em uma data. Caso a nova data cair em um sabado ou domingo será automaticamente transferida para segunda-feira.

Sintaxe:

```
                  <varmem>        <varmem>
PLUS TDATE         DATA1            DIAS
                  <ddmmaa>     <3 posições>

   <varmem>                   <varmem>
     PAR           TO          DATA2
 <1 posição>                  <ddmmaa>
```

Onde:

DATA1: Variável caracter com 6 posições que deve conter a data inferior.

DIAS: Variável caracter com 3 posições que será adicionada em DATA1.

PAR: Variável caracter com 1 posição.

Se igual a "0" somará os dias a partir da data informada (DATA1).

Se igual a "1" somará os dias a partir do primeiro dia do mês seguinte da data informada (DATA1).

DATA2: Variável caracter com 6 posições que retornará com a nova data.

Exemplo:

```
STORE "290885"    TO DATA1
STORE "002"       TO DIAS
STORE "0"         TO PAR
STORE "      "    TO DATA2
PLUS TDATE DATA1 DIAS PAR TO DATA2
? "A NOVA DATA É: " + DATA2
     A NOVA DATA É: 020985
```

CHANGE

Comando que transforma a pontuação do sistema americano para o europeu.

Sintaxe:

	<varmem>	TO	<varmem>
PLUS CHANGE	VALOR1	TO	VALOR2

Onde:

VALOR1: Variável caracter com qualquer tamanho que contém o valor com pontuação americana.

VALOR2: Variável caracter com mesmo tamanho que VALOR1 que retornará o valor com pontuação européia.

Exemplo:

```
STORE "123,634,573.20" TO VLRAM
STORE "                " TO VLREUR
PLUS CHANGE VLRAM TO VLREUR
? VLREUR
123.634.573,20
```

LZERO

Comando que é utilizado para trocar brancos à esquerda por zeros.

Sintaxe:
PLUS LZERO VAR1 TO VAR2

Onde:
VAR1: Variável caracter com qualquer tamanho, que deve conter o número com zeros à esquerda.

VAR2: Variável caracter com mesmo tamanho de VAR1 que retornará com o zero à esquerda.

Exemplos:

```
.STORE "   123" TO VAR1
.PLUS LZERO VAR1 TO VAR2
.? VAR1
. 000123
```

MOD11

Comando que calcula o dígito módulo 11 de um determinado número.

Sintaxe:

	<varmem>		<varmem>
PLUS MOD11	NÚMERO	TO	DÍGITO (1 posição)

Onde:

NÚMERO: Variável caracter com qualquer tamanho que deve conter um número o qual será calculado o dígito de controle módulo 11.

DÍGITO: Variável caracter com 1 posição que retornará com o dígito calculado.

Exemplo:

```
STORE "001" TO NUMERO
STORE "    " TO DIGITO
PLUS MOD11 NUMERO TO DIGITO
? STR(NUMERO)+"-"+STR)DIGITO)
001-9
```

CGC

Comando que verifica validade do dígito do CGC.

Sintaxe:

	<varmem>		<varmem>
PLUS CGC	VAR1	TO	VAR2
	14(posições)		1(posição)

Onde:

VAR1: Variável caracter com 14 posições que deve conter o número do CGC juntamente com o dígito a ser consistido.

VAR2: Variável caracter com 1 posição. Podendo conter qualquer valor inicial. Caso esta variável retorne com valor "1" o dígito é válido, caso contrário ("0") não é um dígito válido.

Exemplo:

```
STORE "6208514700189" TO VRCGC
STORE " " TO RESP
? PLUS CGC VRCGC TO RESP
IF RESP = "1"
? "CGC OK"
ELSE
? "CGC INVÁLIDO"
ENDIF
CGC OK
```

CPF

Comando que verifica a validade do dígito CPF.

Sintaxe:

	<varmem>		<varmem>
PLUS CPF	VAR1	TO	VAR2
	11(posições)		1(posição)

Onde:

VAR1: Variável caracter com 11 posições que deve conter o número do CPF juntamente com o dígito a ser consistido.

VAR2: Variável caracter com 1 posição podendo conter qualquer valor inicial. Caso esta variável retorne com valor "1", o dígito válido, caso contrário ("0") não é um dígito válido.

Exemplo:

```
STORE "08239033890" TO VRCPF
STORE " " TO RESP
PLUS CPF VRCPF TO RESP
IF RESP = "1"
? "DIGITO OK"
ELSE
? "DIGITO INVÁLIDO"
ENDIF
DIGITO OK
```

FORMAT

Comando que monta a máscara de impressão (Formato Europeu).

Sintaxe:

```
              <varmem>     <varmem>          <varmem>
PLUS DDATE     VAR1          VAR2       TO     VAR3
                          1(posição)
```

Onde:

VAR1: Variável caracter que deve conter um número a ser editado.

VAR2: Variável caracter com 1 posição que deve conter a quantidade de casas decimais para a edição.

VAR3: Variável caracter com tamanho mínimo de VAR1 mais a expansão que deve ser prevista devido a colocação de pontos e vírgulas.

Caso não seja prevista a expansão, e o resultado não couber em VAR3, retornará mascarado com "*" indicando estouro de campo.

Exemplos:

```
STORE "12345678" TO NÚMERO
STORE "XXXXXXXXXXXXXX" TO MÁSCARA
STORE "3" TO DEÇIMAIS
PLUS FORMAT NÚMERO DECIMAIS TO MÁSCARA
? MÁSCARA
.12.345,678
```

CODE

Envia a mensagem: "ENTRE COM A SENHA:"

Para a linha e coluna especificadas e aguarda que seja digitada a senha.

Cada caracter digitado será substituído na tela por um ".".

A quantidade de caracteres é determinada pelo tamanho de variável que os receberá.

Sintaxe:
PLUS CODE < **LINHA** > < **COLUNA** > TO < **VARIÁVEL** >

Onde:

< **LINHA** > : Variável tipo caracter com duas posições que indica a linha solicitando a senha.

< **COLUNA** > : Variável tipo caracter com duas posições que indica a coluna solicitando a senha.

< **VARIÁVEL** > : Caracter igual o que já foi digitado.

Exemplo:

```
STORE "02" TO LIN
STORE "10" TO COL
STORE "XXXXX" TO SENHA
PLUS CODE LIN COL TO SENHA
? SENHA
XXXXX
```

DGEN: SUMÁRIO

dGEN torna fácil a criação e execução de programas que facilitam o uso do banco de dados do dBASE II. Para criar os programas, você deve simplesmente responder as perguntas feitas pelo utilitário dGEN.

PROGRAMAS

Os programas dGEN são os seguintes:

(1) dGEN – Menu principal, permite a ativação das opções abaixo mediante o pressionamento de uma tecla. Você pode, porém, ativar qualquer uma das opções abaixo digitando o comando DO, a partir do prompt do dBASE II.

(2) GERAARQ – Cria um menu principal e um conjunto de programas para a edição de um banco de dados existente. As opções do menu permitem a consulta, adição, edição ou compactação dos registros do seu banco de dados.

(3) GERAREL – Cria um menu e programas que fazem relatórios do seu banco de dados. O menu e os relatórios são similares àqueles obtidos usando-se REPORT do dBASE II.

(4) GERAETIQ – Cria um programa de emissão de etiquetas a partir do seu banco de dados. Também pode ser usado para a exibição, em vídeo ou impressora, do conteúdo de registros grandes.

(5) GERAMENU – Cria menus para programas escritos em dBASE II, resultando em uma substancial redução do tempo de programação.

Antes de usar o GERAMENU, os usuários devem estar habilitados a efetuarem pequenas alterações em programas dBASE II, já que os programas gerados irão requerer a adição de algumas linhas de comandos.

PALAVRAS RESERVADAS

As palavras abaixo são reservadas para o dGEN. Não as use em seus nomes de arquivos ou campos.

arqdados	cabec	lin	subcpo
arqindice	cposubtot	largpag	substack
arqsist	cabpag	margeme	subtotal
arqform	contlin	mconteudo	s:n
arqent	expan	Mlarg	seunome
arqsai	expressao	nomearq	selenum
col	extensao	opcao	temsubtot
colcab	escolha	prompt	temtotal
colopcs	field:save	pos	tam
colcont	iguais	palavra	totalopcs
cont	item	selecao	totstack
caract	limpalin	stackcont	ultlinha
cpochave	linha	string	valchave

COMO USAR O dGEN

Procedimentos

O dGEN e os programas por ele criados executam em dBASE II e, trabalham somente com banco de dados criados no dBASE II. Nenhum registro é necessário para iniciar, você pode adicioná-los depois através dos programas de edição criados pelo utilitário GERAARQ.

Criação.DBF

1. Para criar uma duplicata do banco de dados usado nestes exemplos, proceda segundo os itens abaixo.

Observação:
< RETURN > significa "Pressione a tecla < RETURN >".

```
A> DBASE <RETURN>

.CREATE <RETURN>

ENTRE COM O NOME DO ARQUIVO: PESQUISA <RETURN>

ENTRE COM A ESTRUTURA COMO SEGUE:

CAMPO NOME, TIPO, TAMANHO, CASAS, DECIMAIS
001        PARA: INDI, C, 25 <RETURN>
002        DEPARTAMEN, C, 30 <RETURN>
003        COMPANHIA, C, 30 <RETURN>
004        CIDADE, C, 20 <RETURN>
005        ESTADO, C, 2 <RETURN>
006        CEP, C, 5 <RETURN>
007        DATA, C, 5 <RETURN>
008        COMENTARIO, C, 25 <RETURN>
009        <RETURN>

ENTRAR DADOS AGORA? N <RETURN>
```

2. Para todos os utilitários, com exeção do GERAMENU, você necessita dos nomes dos campos do banco de dados, como são colocados no diretório.

   Para obter uma relação a partir do dBASE II, digite os comandos mostrados a seguir. Se você não necessita imprimir, omita os comandos. "set print on" e "set print off"

Lembre-se:
<RETURN> significa, pressione a tecla <RETURN>.

```
.DISPLAY STRUCTURE

ESTRUTURA DO ARQUIVO: PESQUISA.DBF
NUMERO DE REGISTROS: 00000
DATA DA ULTIMA ATUALIZACAO: 01/05/85

USO PRIMARIO DATABASE
CPO NOME              TIPO    TAMANHO    DEC
01      PARA: INDI     C       025
02      DEPARTAMEN     C       030
03      COMPANHIA      C       030
04      RUA            C       025
05      CIDADE         C       020
06      ESTADO         C       002
07      CEP            C       005
08      DATA           C       005
09      COMENTARIO     C       025
** TOTAL **                    00172
```

Acesso

3. Você pode iniciar o dGEN a partir do prompt do sistema operacional (se você já tiver o banco de dados), ou a partir do prompt do dBASE II.

Sistema Operacional	DBASE II	Resultado
A> DBASE DGEN	. DO DGEN	Menu principal
A> DBASE GERAARQ	. DO GERAARQ	Direto ao GERAarq
A> DBASE GERAREL	. DO GERAREL	Direto ao GERArel
A> DBASE GERAETIQ	. DO GERAETIQ	Direto ao GERAetiq
A> DBASE GERAMENU	. DO GERAMENU	Direto ao GERAmenu

Menu Principal

4. Se você digitar "A> DBASE DGEN" ou ". DO DGEN" a seguinte tela será mostrada:

```
DGEN MENU PRINCIPAL
==================================================================
0. fim
1. gerador de MENU
2. gerador de ARQUIVO
3. gerador de RELATORIO
4. gerador de ETIQUETA
======================== selecione :2: ===========================
```

Quando você obtiver este menu, pressione o número correspondente ao utilitário que você deseja – ou 0 (zero) para retornar ao prompt do dBASE II. Cada utilitário do dGEN é explicado na sequência em que aparece no DGEN MENU PRINCIPAL, exceto o gerador de menus – que requer algum conhecimento a mais de programação – que é explicado posteriormente. Os outros utilitários podem ser assimilados rapidamente por um usuário iniciante.

COMO CRIAR PROGRAMAS DE EDIÇÃO

GERAARQ

O utilitário GERAARQ gera um conjunto de programas dBASE II, com os quais você pode consultar, adicionar, editar e compactar registros de um banco de dados classificado. Para criar estes programas, responda, mediante digitação, as perguntas feitas pelo utilitário GERAARQ.

Se for fornecida uma resposta que o GERAARQ não aceite, uma mensagem de erro será exibida, seguida por uma linha em branco e o prompt do dBASE II. Para reiniciar digite: DO GERAARQ <RETURN>.

Acesso

1. O método mais rápido de iniciar o GERAARQ é:

   A) DBASE GERAARQ – a partir do sistema operacional ou . DO GERAARQ – a partir do prompt do dBASE II.

Você pode preferir iniciar a partir do dGEN e selecionar a opção 2 para ativar o GERAARQ. Para iniciar a partir do dGEN, digite: A> DBASE DGEN" ou ". DO DGEN".

Prompts

2. O utilitário GERAARQ exibe o cabeçalho e os primeiros prompts para o programa gerador de "Consulta, Adição, Edição e Compactação".

Observ. da Tela

```
GERADOR (CONSULTA, ADICAO, EDICAO, E COMPACTACAO) DD/MM/AA
================================================================

(a) Digite o nome do arquivo de BANCO DE DADOS: pesquisa
(b) Digite o nome do arquivo de INDICE: pesquisa
(c) Digite o campo chave de INDICE: companhia
(d) ARQUIVO DE INDICE NAO EXISTE. Cria? (S/N): s
(e) Criando arquivo de indice...
    Buscando os nomes dos campos...
(f) Checando a existencia dos nomes de arquivos...
```

OBSERVAÇÕES DA TELA

(a) "Digite o nome de um BANCO DE DADOS existente". No exemplo usamos PESQUISA.

(b) "Digite o nome do arquivo de ÍNDICE". O nome pode ser igual ou diferente do nome do banco de dados.

(c) "Digite o campo chave de índice". Este é o campo ou a expressão pela qual o arquivo está indexado.

(d) Esta pergunta só é exibida quando o arquivo de índice não existe. Digite "S" sempre que ela aparecer, caso contrário nenhuma mensagem de erro irá aparecer e o controle retornará ao prompt do dBASE II. **< Para reiniciar digite ". DO GERAARQ".>**

(e) A mensagem "Criando arquivo de índice..." só irá aparecer quando para a pergunta do item (d) resposta tiver sido "S". A mensagem "Buscando os nomes dos campos..." aparece quando o GERAARQ esta pesquisando os nomes dos campos no diretório.

(f) A mensagem "Checando a existência dos nomes dos arquivos..." aparece enquanto o GERAARQ checa se os nomes dos arquivos dados já existem no diretório. Se ja existirem, será exibida uma pergunta, para que você decida se os arquivos existentes devem ou não ser deletados.

GERAÇÃO:

3. O utilitário GERAARQ gera um conjunto de programas que fornecem as funções de consulta, adição, edição e compactação dos registros do banco de dados PESQUISA. As linhas de codificação do programa são exibidas em vídeo o suficientemente lentas para que você possa ver o que esta sendo feito, ao mesmo tempo em que são escritas e gravadas.

4. Quando o novo programa estiver pronto para ser usado, a seguinte mensagem será exibida:

PARA INICIAR O SISTEMA "PESQUISA", TECLE O SEGUINTE

```
DO PE-PRIN <RETURN>
```

Observações:

No comando ". DO PE-PRIM", as duas letras que antecedem o hífem são as primeiras duas letras do nome do seu banco de dados.

FIM
GERAARQ

5. Você está no prompt do dBASE II podendo, portanto, digitar qualquer comando dBASE II.

COMO USAR PROGRAMAS CRIADOS NO GERAARQ

Um Exemplo

USANDO OS PROGRAMAS

A hora que você quiser usar os programas criados pelo GERAARQ para o banco de dados PESQUISA, proceda segundo os comandos abaixo:

1. Quando no prompt do dBASE II, digite:

```
. DO PE-PRIM <RETURN>
```

para obter a seguinte tela:

```
MENU PRINCIPAL DE PESQUISA
-------------------------------------------------------
0. fim
1. consultar
2. adicionar
3. editar
4. compactar
------------------------ selecione: 2 ------------------
```

Adição de Registros

2. Pressione 2 para adicionar (APPEND) registros ao banco de dados PESQUISA, que até agora não tem nenhum registro. A tela de entrada de dados aparece como a mostrada abaixo:

```
ADICAO PESQUISA                                  DD/MM/AA
-------------------------------------------------------
PARA:     INDI: SR. ALBERTO DE OLIVEIRA <RETURN>
DEPARTAMEN: Departamento Pessoal <RETURN>
COMPANHIA..: MANUFATURAS B&B <RETURN>
RUA...............:                        :
CIDADE ..........:                 :
ESTADO .........:     :
CEP ................:          :
DATA ..............:           :
COMENTARIO .:                              :
-------------------------------------------------------
Pressione <control-W> para sair
```

Preencha cada campo com a sua informação, e pressione <RETURN>. Cada vez que você terminar de digitar todas as informações para o registro, uma nova tela aparecerá para o próximo registro. Quando terminar, mantenha pressionada a tecla <CTRL> e aperte a letra <W> para retornar ao menu principal PE-PRIN. Pressione 0 (zero) se quiser retornar ao prompt do dBASE II: 1 para consultar os registros no banco de dados PESQUISA; 2 para adicionar mais registros; 3 para editá-los.

Já que você não editou nenhum registro, nenhum foi marcado para deleção e, portanto, a opção 4 do compactação não pode ser escolhida. Selecione a opção 3 para editar os registros que Registros já foram incluídos no banco de dados.

O primeiro registro exibido na tela é o seguinte:

```
EDICAO PESQUISA                                          DD/MM/AA
-----------------------------------------------------------------
PARA:     INDI: SR. Alberto de Oliveira
DEPARTAMEN: Departamento Pessoal
COMPANHIA .: MANUFATURAS B&B
RUA .............: R. Becker, 344
CIDADE .........: Sao Paulo
ESTADO ........: SP
CEP ..............: 01311
DATA ............: 05/01
COMENTARIO .: Gerente Departamental

Comando: (E)ditar (D)eletar (R)ecuperar (C)ontinuar (P)osicionar
```

A linha de prompt para EDIÇÃO está repetida para uma melhor referência.

COMANDO: (E)ditar (D)eletar (R)ecuperar (C)ontinuar (P)osicionar

E – Editar. Permite a correção de qualquer campo do registro, exceto o campo chave, que é mostrado em meia intensidade. Se o campo chave esta incorreto, delete o registro e use a opção 2 de adição de um novo registro.

D – Deletar. A mensagem "DELETADO" aparece à esquerda da data no alto da tela. Você deve executar a opção 4, COMPACTAR, para apagar fisicamente o registro.

R – Recuperar. Apaga a mensagem "DELETADO"

C – Continuar. Se você não editou, deletou ou recuperou o registro corrente, esta traz a você o próximo registro.

P – Posicionar. Traz a você uma nova "linha de prompt" como a abaixo:

```
EDICAO PESQUISA                                              DD/MM/AA
------------------------------------------------------------------------
PARA:           INDI: Sr. Alberto de Oliveira
DEPARTAMEN: Departamento Pessoal
COMPANHIA .: MANUFATURAS B&B
RUA ............: Rua Becker, 344
CIDADE .........: Sao Paulo
ESTADO ........: SP
CEP ..............: 01311
DATA ...........: 05/01
COMENTARIO : Gerente Departamental

COMANDO: (M)ostrar (P)esquisar (L)ocalizar (C)ontinuar prompt (A)vancar
```

Para obter a segunda linha de prompt da EDIÇÃO, selecione a opção P, da primeira tela. A linha de prompt será repetida para uma melhor referência.

COMANDO: (M)ostrar (P)esquisar (L)ocalizar (C)ontinuar (A)vancar

M – Mostrar. Permite consultar mais que um registro por vez baseado em uma expressão LOCATE. Digite a expressão para localização, como por exemplo:

COMPANHIA = MANUFATURAS B&B

Depois digite o "string" a ser exibido. Por exemplo: Para: indi + Companhia

P – Pesquisar. Digite o texto da palavra chave (COMPANHIA) para o registro desejado. Por exemplo:

MANUFATURAS B&B

L – Localizar. Baseado na expressão LOCATE, de maneira similar a opção (M)ostrar.

C – Continuar. Avança para o próximo registro. Ao final do arquivo pressione <RETURN>, para retornar ao prompt COMANDO.

A – Avançar. Avança para o próximo registro. Ao final do arquivo pressione <RETURN>, para retornar ao prompt COMANDO.

Consulta

```
CONSULTA PESQUISA                                              DD/MM/AA
-----------------------------------------------------------------------
```

4. Selecione a opção 1 para "CONSULTAR" qualquer registro do banco de dados. A tela de consulta é idêntica a tela de edição, com exceção do cabeçalho que apresenta CONSULTA ao invés de EDIÇÃO.

A opção de CONSULTA opera da mesma maneira que a segunda tela da EDIÇÃO, com a diferença que a CONSULTA não permite alteração dos registros.

Compactação de registros

```
COMPACTACAO PESQUISA                                           DD/MM/AA
-----------------------------------------------------------------------
Compacta o arquivo inteiro? (SIM/NAO) SIM <RETURN>
PESQUISA.ANT sera seu arquivo back-up.
COPY TO PESQUISA
004 REGISTROS INDEXADOS...
USE
SET ECHO OFF
Aperte qualquer tecla para continuar...
```

5. Selecione a opção 4 para deletar definitivamente os registros marcados para deleção na primeira tela da opção de EDIÇÃO.

Quando for perguntado se você quer COMPACTAR o arquivo inteiro, você deve digitar SIM ou NÃO.
Se digitar NÃO, retornará ao Menu Principal para edição.
Se digitar SIM, um arquivo de back-up é criado com os dados antes da compactação.
Aperte qualquer tecla para retornar ao Menu Principal.

PROGRAMA GERADOR DE RELATÓRIO

GERAREL

O utilitário GERAREL, cria um arquivo de comandos que a partir de um banco de dados existente gera um relatório similar aqueles criados pelo comando REPORT do dBASE II. A saída pode ser em impressora ou vídeo. Abaixo esta uma breve descrição de criação de um relatório a partir do banco de dados PESQUISA.

EMPREGO	EMPREGOS OFERECIDOS NOME DA COMPANHIA	DATA
Gerente de operacao	Bandeirante	3/7
Planejamento financ	Aviacao Ltda.	3/8
Consultar manufat.	Silva & Costa	3/10
Gerente de Producao	Manufatura B&B	3/5
Gerente de Projetos	C & C Consultores	3/3
Gerente de Producao	Sistemas Integrados TR	3/14
Mg – Planejamento	Sistemas Digitais S/C	2/25
Supervisor Producao	Plasticos Plavitol	2/15

OBSERVAÇÃO

A saída acima especificada pode ser mais útil se antes de ser impressa for classificada por COMENTÁRIO ou COMPANHIA.

PROCEDIMENTOS

Para criar e usar um programa de relatório, proceda segundo as instruções que seguem:

Acesso

1. O acesso ao utilitário GERAREL pode ser tanto feito diretamente do prompt do sistema operacional como do dBASE II:

   A>DBASE GERAREL ou . DO GERAREL

   Se você preferir, pode acessá-lo a partir do Menu Principal dGEN selecionando a opção 3.

Prompts

2. O utilitário GERAREL, exibe uma ou duas das perguntas mostradas abaixo, por vez. As respostas aplicáveis ao banco de dados PESQUISA estão em negrito.

Comentários da Tela

```
(a) Digite o nome do arquivo de BANCO DE DADOS:
    PESQUISA <RETURN>

(b) Digite o nome do arquivo de formato de relatorio:
    EMPREGOS <RETURN>

        DIGITE AS OPCOES

(c) Margem esq... <1>: 10 <RETURN>
    Linhas/pag.. <56>: <RETURN>
    Largura Pag. <80>: 55 <RETURN>

Digite Cabeçalhos da Pagina, EMPREGOS OFERECIDOS <RETURN>
Totais (S/N): N <RETURN>
Subtotais no Relatorio? (S/N): N <RETURN>

DIGITE A DESCRICAO DAS COLUNAS:
1.   Tamanho, Conteudo: 20, COMENTARIO <RETURN>
Cabecalho..........: EMPREGOS <RETURN>
2.   Tamanho, Conteudo: 20, COMPANHIA <RETURN>
Cabecalho .........: NOME DA COMPANHIA <RETURN>
3.   Tamanho, Conteudo: 8, DATA <RETURN>
Cabecalho .........: DATA <RETURN>
4.   Tamanho, Conteudo: <RETURN>
```

Comentários da Tela

(a) Digite o nome do arquivo de BANCO DE DADOS, do qual você quer o relatório.

(b) Digite o nome do arquivo de relatório. Este é o nome que usará quando quiser criar o relatório.

(c) DIGITE AS OPÇÕES para a margem esquerda, linhas/página e largura da página. Para assumir os valores default pressione <RETURN>, caso contrário digite os valores e pressione <RETURN>

(d) Digite o Cabeçalho da página – digitar qualquer conjunto de caracteres que possa ser colocado em uma linha. O cabeçalho é centralizado dentro da largura especificada acima.

(e) Totais e subtotais. Totais? Subtotais? Se "S", GERAREL cria a programação necessária para cada coluna, como requerido. Você pode escolher qualquer campo do banco de dados para subtotalizar. Caso o campo que você digitou não for encontrado no diretório do banco de dados a mensagem "Digite o campo de subtotal" será novamente exibida.

(f) Digite a descrição das colunas. Você usa esta pergunta para especificar a largura da coluna, a descrição dos campos e o cabeçalho da coluna (opcional).

Largura: Se a lagura da coluna que você especificar for menor que o tamanho do campo do banco de dados, a informação na coluna será truncada.

Próxima coluna: Depois que cada coluna for descrita, um novo número de coluna é exibido para que as próximas descrições possam ser digitadas. Após você ter digitado todas as colunas necessárias, pressione <RETURN>.

OBSERVAÇÃO

É permitido um máximo de doze (12) colunas descritivas de campos. Lembre-se, no entanto, que os campos podem ser agrupados através de algumas funções do dBASE II.

Criação

3. O utilitátio GERAREL inicia a gravação do programa que cria o relatório requerido. Os comandos são exibidos na tela enquanto a gravação é feita, de modo muito rápido para serem lidos, mas lentos o suficiente para se entender o que esta sendo feito.

Saída

4. Para imprimir o novo, relatório – na tela ou impressora – você deve ir ao dBASE II (pressione a opção 0 (zero) para sair). Quando no prompt do dBASE II, digite "DO <nome do arquivo de relatório>", onde <nome do arquivo de relatório> é o nome que você digitou no passo 2. No nosso exemplo digitaremos:

```
. DO EMPREGOS <RETURN>
```

O programa pergunta se você quer a saída em tela (T) ou impressora (I). Digite a sua escolha, e o programa iniciará a execução imediatamente. Ao final da execução o controle será retornado ao prompt do dBASE II.

PROGRAMA GERADOR DE ETIQUETAS

GERAETIQ

O utilitário GERAETIQ cria um programa que a partir dos dados de um arquivo de banco de dados existente desenha e imprime etiquetas para mala direita. Abaixo estão três etiquetas impressas por um programa gerado pelo GERAETIQ e que usa os dados do arquivo PESQUISA.

```
1   Sr. Alberto de Oliveira
2   Departamento de Recursos Humanos
3   R. Becker, 334
4   CEP – 01311
5   SP
-
1   Caixa Postal 15
2   05020 Sao Paulo – SP
3
4
5
-
1   Srta. Claudia Lemos
2   Sistemas Digitais S/C
3   Caixa Postal 234
4   05122 Sao Paulo – SP
5
```

Outro uso GERAETIQ

O banco de dados PESQUISA tem registros grandes com nove campos – muita informação para colocar em único relatório de 80 ou 132 colunas. O utilitário GERAETIQ permite que você crie um programa para impressão do conteúdo completo de cada registro em um formato fácil. Para isto use o GERAETIQ de modo normal, mas peça todos os campos desejados (na ordem que quiser) e os imprima em formulário contínuo ao invés de usar formulário com etiquetas.

PROCEDIMENTOS

Para criar e usar o programa de emissão de etiquetas para mala-direta, proceda segundo as especificações a seguir:

Acesso

1. Ative o GERAETIQ diretamente a partir do sistema operacional ou do prompt do dBASE II:

A>DBASE GERAETIQ ou . DO GERAETIQ

Se você preferir, obtenha o GERAETIQ a partir do menu principal do dGEN selecionando a opção 4.

Prompts

2. Os prompts seguintes serão exibidos quando o utilitário for acessado: as respostas aplicáveis ao PESQUISA estão em negrito.

```
Procedimentos:

(a)  Digite o nome do arquivo de BANCO DE DADOS:
     PESQUISA <RETURN>
(b)  Digite o nome do arquivo de ETIQUETAS:
     LISTAEMPR <RETURN>
(c)  Digite as linhas da ETIQUETA (exemplos):
     1: NOME
     2: [COMPANHIA] <-- imprimira somente quando nao branco
     3: ENDEREÇO
     4: TRIM(CIDADE) +",  "+ESTADO+" "+CEP
     5:
(d)  Digite as linhas das ETIQUETAS:
     1: PARA: INDI <RETURN>
     2: [DEPARTAMEN] <RETURN>
     3: COMPANHIA <RETURN>
     4: RUA <RETURN>
     5: TRIM (CIDADE) +" "+ESTADO+" CEP <RETURN>
(e)  6: <RETURN>
```

Observações:

(a) Digite o nome do arquivo de banco de dados. Se o arquivo que você especificar não for encontrado, a tela será limpa e a seguinte mensagem será exibida "Aperte qualquer tecla para continuar..." Quando você pressionar a tecla. Certifique-se, portanto, do nome que irá fornecer, bem como o drive se este for diferente do drive default.

(b) Digite o nome do arquivo de ETIQUETA, que é o nome do arquivo de comandos que irá identificar este formato de etiqueta em particular, que criará etiquetas somente para as informações do banco de dados referido acima.

(c) Digite as linhas de ETIQUETA (exemplo): mostra o que deve ser digitado para produzir quatro linhas de etiqueta. Este exemplo aparecerá em sua tela em tempo de execução, um pouco acima do local onde você deverá digitar as suas linhas de etiqueta.

(d) Digite as linhas da ETIQUETA. Este é o local onde você realmente deve digitar as informações que o GERAETIQ necessita para criar o seu programa gerador de etiquetas. Qualquer campo inserido entre colchetes, somente será impresso se houver algum conteúdo. Você pode usar as opções que estão em negrito para criar etiquetas para o arquivo de banco de dados PESQUISA.

(e) Quando terminar de digitar todas as linhas da etiqueta, pressione <RETURN>.

Criação

3. O utilitário GERAETIQ gera um arquivo de comandos que cria as etiquetas. As linhas de comando são listadas em vídeo enquanto o programa é gerado.

   Quando o novo programa estiver pronto para ser usado, GERAETIQ retornará o controle ou para o prompt do dBASE II, se você o iniciou a partir do dBASE II, ou para o dGEN menu principal se o iniciou a partir da seleção 4 do Menu dGEN.

Saída

4. Para imprimir as etiquetas da mala-direta na tela ou impressora você deve estar no prompt do dBASE II.

   Quando no prompt, digite "DO <nome de etiquetas>", onde <nome do arquivo de etiquetas> é o nome do programa. Em nosso exemplo, o comando seria:

```
.DO LISTEMPR <RETURN>
```

5. A tela "MALA DIRETA" será exibida com uma pergunta, como mostrada abaixo: ou <RETURN> para cancelar o programa e retornar ao prompt do dBASE II.

PESQUISA MALA DIREITA	DD/MM/AA
Saida para tela ou impressora? [S/N]	

COMO CRIAR MENU PRINCIPAL

GERAMENU

O utilitário GERAMENU cria programas para você. Por exemplo, o menu principal dGEN foi criado pelo GERAMENU. Você pode usá-lo para criar menus como os mostrados a seguir:

Cabecalho o menu	PESQUISA MENU PRINCIPAL
Opcoes **linha de mensagem**	0. fim 1. Consulta, Edicao, Delecao ou Compactacao 2. Impressao do relatorio LISTAEMPR 3. Impressao de MALA DIRETA
....selecione:..	

Cabecalho **do menu**	JOGOS MENU PRINCIPAL
Opcoes **linha de mensagem**	0. fim 1. Jogo simulacao de voo 2. Jogo batalha final 3. Jogo smurph ....selecione: ....

Importante

Depois de criar o menu, você pode exibí-lo ativando o programa através de um DO. Todavia, se você selecionar qualquer opção, a tela irá piscar e o menu será novamente exibido.

Para ter seu menu trabalhando de modo correto você necessita de duas coisas: programas que executem as opções do menu e pequenas alterações no programa gerado pelo GERAMENU.

Para aprender como fazer estas modificações leia a página 9-22 "Somente para programadores".

Requisitos

Você não necessita de um banco de dados para criar um menu usando o GERAMENU. Nem mesmo necessita dos programas que executem as opções do menu. Necessita somente de uma lista de funções (opções) que queira que o menu exiba.

Procedimentos

Para criar e usar um menu principal para quaisquer programas em dBASE II, você deve fazer o seguinte:

Acesso

1. O modo mais rápido de se ativar o GERAMENU é:

   A>DBASE GERAMENU – a partir do sistema operacional
   .DO GERAMENU – a partir do prompt do dBASE II

   Se preferir, ative o menu principal dGEN e selecione a opção 1 para iniciar o GERAMENU. Para ativar o menu principal dGEN, digite:

   A>DBASE DGEN   ou
   . DO DGEN

Prompts

2. O utilitário GERAMENU exibe a tela baixo, as respostas em negrito criam o programa que pode ser usado para exibir o menu para o banco de dados PESQUISA.

```
Numero do

Comentario

da Tela

(a) Digite o nome do PROGRAMA: PESQEMPR <RETURN>

(b) Digite o cabecalho do MENU: Pesquisa de Emprego <RETURN>

(c) Digite as opcoes do menu:

      0. fim
      1. Consulta, Edicao, Delecao de registros <RETURN>
      2. Impressao do relatorio LISTAEMPR <RETURN>
      3. Impressao da MALA DIRETA <RETURN>
      4. <RETURN>

(d) COMANDO: (S)air, (R)efazer, (G)ravar
```

Comentários da Tela

(a) Use o nome do programa para chamar o menu principal quando o programa estiver pronto para uso.

(b) O cabeçalho do menu é expandido e exibido no topo da tela do novo menu principal. Poderemos chamar o nosso menu principal de pesquisa de emprego ativando o comando "DO PESQEMPR".

(c) Digite as opções do menu. A opção 0 (zero) é sempre "fim" – a menos que você modifique o programa GERAMENU ou o programa gerado – opção que retorna o controle ao prompt do dBASE. Digite a opção 1, e em seguida pressione <RETURN> para ir a próxima opção. O limite máximo é de 14 opções. Quando terminar, pressione G <RETURN>.

(d) COMANDO: (S)air, (R)efazer, (G)ravar

Onde:

(S)air – Retorna ao prompt do dBASE II ou para o dGEN sem criar o programa de menu.
(R)efazer – Repete todo o processo desde o início (nome do programa e cabeçalho). Você reinicia o processo.
(G)ravar – O GERAMENU gera o programa de menu principal de acordo com as informações fornecidas.

Geração do Programa

3. Quando você seleciona a opção (G)ravar, o programa que exibe o menu principal é criado e, durante a sua criação, exibido na tela.

4. Quando o novo programa de menu principal está pronto para uso, o controle será retornado ou para o prompt do dBASE II ou para o menu principal dGEN, conforme tenha sido o seu modo de ativação do GERAMENU.

Consulta ao menu

5. Para ativar o novo menu principal, digite: "DO <nome do programa>", neste caso:

```
.DO PESQEMPR
```

SOMENTE PARA PROGRAMADORES

Criando

Os programas de menu principal que você criou com o GERAMENU, não funcionarão até você modificá-los. Use o MODIFY COMMAND do dBASE II para chamar os programas.

1. Vá a instrução "DO CASE". Observe que cada opção do menu principal está descrita em um "CASE".

2. Agora, insira uma nova linha entre cada comando "CASE" e acrescente o comando "DO <nome do arquivo de programa>". Por exemplo:

   ```
   . DO PESQEMPR
   ```

3. Enquanto estiver no MODIFY COMMAND, você também pode:

(a) Colocar o seu nome no início do programa, na opção AUTOR e,

(b) Fazer qualquer modificação desejada no texto do programa. Por exemplo no texto da opção o trocar "fim" por "retornar ao prompt do dBASE II".

dGEN

O utilitário GERAARQ usa o seguinte formato de nome de programas para criar os seus programas.

```
DD-PROGRAMA.EXT
```

Onde:
DD = duas primeiras letras do nome do banco de dados. PE para PESQUISA.

PROGRAMA = PRIN, TELAS, QUALQ, GETS, EDIT, PACK, PESQ E LOCAT; e

EXT = a extensão do programa; .PRG